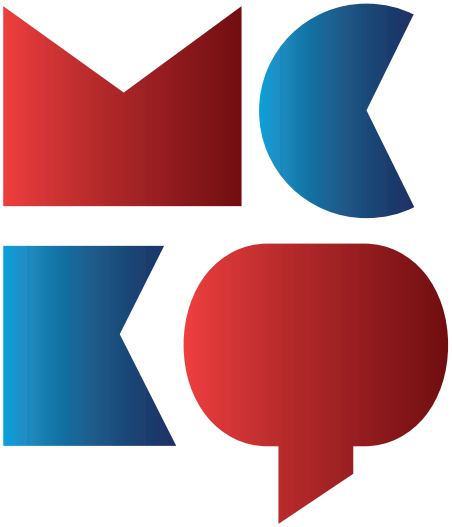
РОССИЙСКАЯ АКАДЕМИЯ НАУК

РОССИЙСКИЙ ФОНД ФУНДАМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ

ЗАО «ОТКРЫТЫЕ СИСТЕМЫ»



ЧЕТВЕРТЫЙ МОСКОВСКИЙ СУПЕРКОМПЬЮТЕРНЫЙ ФОРУМ

Тезисы докладов

Москва, 2013 год

УДК 519.6, 519.7, 004.2, 004.3

ББК 32.973

Оргкомитет Четвертого Московского суперкомпьютерного форума выражает признательность за поддержку Российскому фонду фундаментальных исследований (грант 13-07-06046-г), Отделению нанотехнологий и информационных технологий РАН, компаниям «РСК», «Т-Платформы», Intel и NVIDIA.

Тематика МСКФ-2013:

Пленарная сессия. Перспективы экзамасштабных систем

Секция. Экзафлопсные архитектуры

Секция. Практика и кадры суперкомпьютерной индустрии

Секция. Стендовые доклады

Тезисы докладов Четвертого Московского суперкомпьютерного форума (Москва, 23 октября 2013 г.). / [Под. ред. Волкова Д.В.]. — М.: «Открытые системы», 2013. — 48 с.

В сборник трудов включены доклады Четвертого Московского суперкомпьютерного форума, прошедшего 23 октября 2013 года в ЦВК «Экспоцентр» на Красной Пресне. Целью форума была консолидация усилий научно-исследовательских и промышленных коллективов, создающих элементно-конструкторскую базу экзамасштабных высокопроизводительных систем и программные средства решения задач в параллельных средах. Материалы сборника тезисов предназначены для научных сотрудников, разработчиков, преподавателей, аспирантов и студентов, интересующихся проблемами создания и эксплуатации суперкомпьютерных систем экзафлопсного уровня производительности. Подробную информацию о МСКФ-2013 можно найти по адресу www.ospcon.ru.

Суперкомпьютерное будущее России

Государство, которое не хочет кормить свою науку, работающую для развития отечественных высоких технологий, будет вынуждено обслуживать чужие технологии, что непосредственно относится и к индустрии высокопроизводительных систем. Наличие сырьевых ресурсов оказывает сегодня все меньшее влияние на положение государства на мировой арене — ключевыми факторами успеха становятся обладание технологиями работы с Большими Данными, наличие у государства средств прогнозной аналитики реального времени и собственных инфраструктур поддержки облаков. Уже начиная с 2007 года во всем мире производится данных больше, чем человечество в состоянии хранить на всевозможных носителях, поэтому парадигма «запомнил — обработал» уже не работает: сегодня требуется подход «обработал — запомнил». Государства и компании, игнорирующие эти изменения и не работающие над созданием собственных систем экзафлопсной производительности, способных «на лету» обрабатывать потоки данных из разнообразных источников, неизбежно проиграют. Четвертый Московский суперкомпьютерный форум показал, что в России имеются технологии, необходимые для создания экзамасштабных систем, и дееспособные команды разработчиков, а у страны еще есть шансы остаться в клубе суперкомпьютерных держав.

Какими бы изощренными ни были традиционные компьютеры, они остаются пока инструментами прошлого века — для решения современных задач создания лекарств, нейтрализации террористических угроз, поддержки гибких облачных сред, аналитики реального времени или обработки Больших Данных требуются новые архитектуры. На Четвертом Московском суперкомпьютерном форуме обсуждались перспективные технологии создания экзамасштабных компьютеров, новационные разработки постмуровской эпохи, а также особенности применения высокопроизводительных систем в различных отраслях национальной экономики, начиная от мобильных решений и кончая глобальными аналитическими системами. Программа форума включала более 30 докладов по всем основным направлениям развития суперкомпьютерной индустрии — сегодня можно с уверенностью утверждать, что МСКФ стал площадкой для сообщества специалистов, компаний-производителей и пользователей, применяющих высокопроизводительные системы в различных отраслях национальной экономики.

МСКФ-2013 стал очередным шагом к консолидации работ отечественных специалистов, опирающихся на национальные научно-технические разработки при выполнении проектов, соответствующих мировому уровню. Согласно единодушному мнению участников Четвертого МСКФ, создание отечественного экзафлопсного суперкомпьютера, на 3–4 порядка превышающего по производительности современные системы, должно стать задачей национального масштаба и решаться на федеральном уровне с привлечением всех существующих в России команд разработчиков, в противном случае имеющийся сегодня научно-технический потенциал в этой области останется невостребованным.

Организационный комитет МСКФ-2013

Организационный комитет:

Гуляев Ю. В.	академик РАН, член президиума РАН, директор ИРЭ им. В. А. Котельникова РАН, председатель оргкомитета
Волков Д. В.	с.н.с. ИПМ им. М. В. Келдыша РАН, гл. редактор журнала «Открытые системы. СУБД», зам. председателя оргкомитета
Бетелин В. Б.	академик РАН, директор НИИСИ РАН
Воеводин В. В.	чл.-корр. РАН, зам. директора НИВЦ МГУ
Иванников В. П.	академик РАН, директор Института системного программирования РАН
Каляев И. А.	чл.-корр. РАН, директор НИИ МВС
Левин В. К.	академик РАН, научный руководитель ФГУП «НИИ "Квант"»
Соловьев В. П.	д.физ.-мат.н., первый зам. директора ФГУП «РФЯЦ-ВНИИЭФ», директор ИТМФ
Савин Г. И.	академик РАН, директор МСЦ РАН
Стемпковский А. Л.	академик РАН, директор Института проблем проектирования в микроэлектронике (ИППМ)
Четверушкин Б. Н.	академик РАН, директор ИПМ им. М. В. Келдыша РАН

Программный комитет:

Волконский В. Ю.	к.т.н., ОАО «ИНЭУМ им. И. С. Брука»
Корнеев В. В.	д.т.н., зам. директора по науке ФГУП «НИИ "Квант"»
Кузьминский М. Б.	к.хим.н., зам. зав. лаб. ИОХ РАН
Слуцкин А. И.	к.т.н., с.н.с., ОАО «НИЦЭВТ»
Сухомлин В. А.	д.т.н., профессор, ВМК МГУ
Федоров А. Р.	к.т.н., директор ООО «ИТ-экспо»
Христов П. В.	к.физ.-мат.н., вице-президент ЗАО «Открытые системы»
Черепенин В. А.	чл.-корр. РАН, зам. директора ИРЭ им. В. А. Котельникова РАН
Шагалиев Р. М.	д.физ.-мат.н., первый зам. директора ИТМФ ФГУП «РФЯЦ-ВНИИЭФ»
Эйсымонт Л. К.	к.физ.-м.н., ФГУП «НИИ "Квант"»

Пленарная сессия. Перспективы экзамасштабных систем

Проекты экзафлопсных суперкомпьютеров за рубежом и в России, ограничения и перспективы роста

Горбунов В. С., Эйсымонт Л. К., Елизаров Г. С. — НИИ «Квант»

В докладе рассматривается реализация некоторых решений создания экзафлопсных суперкомпьютеров на таких архитектурных принципах как: иерархичность, гетерогенность и гибридность, которые уже проявились в суперкомпьютерах Tianhe-2, Cray XC30 и IBM Power755. Обсуждаются преимущества, обеспечиваемые применением таких принципов, а также их возможное усиление в будущем за счет внедрения новейших технологий нано- и микроэлектроники, нанофотоники, матриц микролазеров и микролинз, оптоволоконных соединений, новых 3D-конструктивов и соединений кристаллов в них типа TSV.

Использование перспективных аппаратно-программных принципов и новых технологий в рамках традиционных подходов к созданию суперкомпьютеров не может безгранично улучшать их характеристики, поскольку уже сказываются физические ограничения: производительность до 32–128 EFLOPS («точка Стерлинга») для обычных нереверсивных суперкомпьютеров (ограничение Лэндауэра) и возможности их преодоления (принцип Неймана — Лэндауэра).

По мнению авторов доклада, современный этап исследований и разработок в области экзафлопсных суперкомпьютеров на примере США характерен тем, что одновременно ведутся работы по следующим трем основным направлениям:

1. Внедряются результаты программы DARPA HPCS (2002–2010 годы) в коммерческих образцах суперкомпьютеров, например в Tianhe-2, Cray XC30 и IBM Power755, а также заказных субэкзафлопсных суперкомпьютерах для решения некоторых важных задач (2013–2017 годы).

2. Выполняются программы по созданию перспективных суперкомпьютеров экзафлопсного и более уровня производительности, например программа DARPA UHPC (2010–2020 годы) и программы Министерства энергетики США (DoE).

3. Ведутся работы по оптимизации использования имеющихся КМОП-технологий, а также по новой элементной базе постмуровской эры — например, выполняется программа DARPA STARNet (2013–2028 годы), в рамках которой создана сеть центров прорывных исследований, а также программа IARPA C3 (2013–2018 годы) создания прототипа суперкомпьютера на сверхпроводниковой электронике одноквантовой логики (SFQ).

Планы создания образцов перспективных суперкомпьютеров:

- 2015–2017 годы — заказные суперкомпьютеры, оптимизированные как для решения задач с хорошей пространственно-временной локализацией работы с памятью (CF-задачи), так и задач с плохой пространственно-временной локализацией работы с памятью (DIS-задачи);
- 2018–2020 годы — создание в лабораториях ядерно-оружейного комплекса DoE (NNSA) экзафлопсного суперкомпьютера эволюционного типа путем улучшения существующих суперкомпьютеров и их экстенсивного развития;
- после 2022 года — создание в лабораториях исследовательского комплекса DoE (OS/ASCR) экзафлопсного суперкомпьютера с высокими характеристиками по реальной производительности на широком классе задач, а также по энергопотреблению, надежности и продуктивности программирования;
- после 2020 года — создание заказных специализированных суперкомпьютеров уровней зетта (2020 год) и йотта (2024 год), в которых будут применены такие технологии постмуровской эры, как RSFQ, QCA и квантовые аналогово-спиновые.

О перспективных архитектурных принципах экзафлопсных суперкомпьютеров будущего можно судить по работам [1–3].

Характерная общая черта новейших суперкомпьютеров — иерархическая архитектура, обусловленная применением иерархической коммуникационной сети. Главный элемент такой сети — одночиповый маршрутизатор с множеством связей (линков). Например, в IBM Power 775 это IBM HUB Chip с четырьмя процессорными интерфейсами (суммарная односторонняя пропускная способность 768 Гбит/с) и с 47 сетевыми линками (суммарная пропускная способность 4704 Гбит/с). Сетевые линки обеспечивают соединения вычислительных узлов на трех уровнях иерархии, а маршрутизатор при этом реализует не только передачу сообщений на том или ином уровне, но и передачи с одного уровня на другой. Применение таких сетей вместо применявшихся ранее сетей с топологиями жирного дерева и N-тор в разы повышает реальную производительность суперкомпьютеров, особенно на задачах, для которых требуется использование глобально адресуемой памяти большого объема.

Гибридность — неоднородность на уровне процессорных ядер, а гетерогенность — неоднородность, на более высоком уровне, когда в состав суперкомпьютера входят целые сегменты из специализированных процессоров и сетей, оптимизированных для решения какой-либо задачи или класса задач. В некоторых суперкомпьютерах Cray примером такого неоднородного сегмента был массово-мультитредовый суперкомпьютер Cray XMT, а в Tianhe-2 имеется сегмент на базе 4096 256-тредовых микропроцессоров FT-1500, ориентированных на решение информационно-аналитических задач и выполнение сервисных функций ввода-вывода. Ранее в системе Tianhe-1A такой сегмент содержал 2048 64-тредовых микропроцессоров FT-1000, то есть очевидна тенденция усиления этого свойства. Гетерогенность позволяет за счет специализации резко повысить производительность, особенно важно проявление гетерогенности в виде специализированных процессоров на элементной базе нового поколения. Знаменательным примером такого явления служит специализированный 128-кубитовый аналогово-спиновый квантовый суперкомпьютер канадской компании D-Wave. Один процессор такого суперкомпьютера при решении определенной задачи оптимизации на 4–6 порядков быстрее серверного узла на базе микропроцессоров Xeon [1].

Гибридность может быть применена для решения достаточно большого набора разных задач. Неоднородные ядра могут быть в разных микропроцессорах вычислительного узла,

а могут находиться и внутри одного микропроцессора. В случае Tianhe-2 вычислительный узел — это два 12-ядерных микропроцессора Xeon Ivy Bridge (2,2 ГГц, пиковая производительность 211,2 GFLOPS, в каждом ядре два треда) и три 57-ядерных Xeon Phi (1,1 ГГц, 1,003 TFLOPS, технология 22 нм, в каждом ядре 4 треда). Оперативная память вычислительного узла — 88 Гбайт, из которых 64 Гбайт — память Ivy Bridge, а 24 Гбайт — память Xeon Phi. Такое массовое применение Xeon Phi — уникальное явление, как и сам микропроцессор, поскольку он содержит увеличенное количество ядер и много тредов, это, по-видимому, основной тренд в развитии микропроцессоров. В связи с этим было проведено его углубленное исследование, результаты которого описаны в [2] и были представлены в докладе.

Варианты дальнейшего развития микропроцессоров типа Xeon Phi хорошо видны на примере известных проектов создания перспективных базовых микропроцессоров для экзафлопсных систем с уровнем производительности 10–20 TFLOPS: Echelon (NVIDIA/Cray), Corona (HP). Первый проект отличается гибридностью ядер, а второй — интенсивным применением нано- и микроэлектронных технологий.

Есть ли физические ограничения на увеличение производительности суперкомпьютеров существующей фоннеймановской архитектуры? Этому вопросу посвящена часть доклада, связанная с особенностями создания элементной базы постмуровской эры.

На сегодняшний день полупроводниковая промышленность освоила КМОП-технологию 22 нм, а компания Intel намеревается достигнуть 5 нм к 2020 году. По мнению аналитиков, дальнейшая миниатюризация уже невозможна и закон Мура перестанет работать не позднее 2024 года. Развитие новых технологий логических элементов и элементов памяти упирается *в ограничение Лэндауэра,* которое может быть преодолено только согласно *принципу Неймана — Лэндауэра*. Из ограничения Лэндауэра выводится условное ограничение производительности в 32–128 EFLOPS для суперкомпьютеров на известных сегодня технологиях. Принцип Неймана — Лэндауэра формулирует условие преодоления ограничения Лэндауэра: в системе с логической обратимостью можно добиться и физической обратимости. Смысл принципа в том, что если можно избавиться от потери битов при обработке информации, то затрачивать на обработку одного бита можно много меньше определенной величины. Попытки реализации этого принципа сегодня делаются в реверсивных машинах. Именно применение этого принципа, а также специализированных процессоров, работающих даже на аналоговых принципах, должно обеспечить выход на уровни производительности зетта и йотта.

Сегодня активно внедряются результаты зарубежных исследований прошлого десятилетия по петафлопсной тематике, в которых решалась фундаментальная проблема преодоления неэффективности работы с памятью (проблема «стены памяти»). За рубежом интенсифицируются исследования по экзамасштабной тематике и новой элементной базе постмуровской эры. Важнейшие направления системных исследований и разработок: иерархические сети, гибридные массово-мультитредовые многоядерные микропроцессоры, гетерогенность. Важнейшие направления по элементно-конструкторской базе: нанофотоника, сверхпроводниковая электроника, квантовая электроника, TSV-соединения, оптоволоконные соединения через матрицы линз и лазеров. Новая элементно-конструкторская база позволит преодолеть зетта- и йоттауровень производительности, это намечено в первую очередь в заказных специализированных суперкомпьютерах.

В России имеется Концепция создания экзафлопсных систем, но она пока не стала федеральной программой. Ведется множество отдельных проектов — например, проект МГВС (ИПМ им. М. В. Келдыша РАН, ФГУП НИИ «Квант», физфак МГУ), нацеленный на поддержку в

мощной установке с множеством ПЛИС ряда проектов по имитационному моделированию архитектур и нового программного обеспечения.

http://www.ospcon.ru/files/media/Eysymont.pdf

Литература

1. Горбунов В., Елизаров Г., Эйсымонт Л. Экзафлопсные суперкомпьютеры: достижения и перспективы // Открытые системы. 2013. № 7. С. 10-15.

2. Андрюшин Д., Горбунов В., Эйсымонт Л. Перспективные особенности Tianhe-2 // Открытые системы. 2013. № 8. С. 20-25.

3. Горбунов В., Эйсымонт Л. Комплексная методика тестирования производительности суперкомпьютеров, профессиональный подход. Вычислительные методы и программирование. 2013 (в печати).

Экзафлопсные вычисления: модели, алгоритмы, программные комплексы

Четверушкин Б. Н., Якобовский М. В. — ИПМ им. М. В. Келдыша РАН

Создание методов, обеспечивающих эффективное использование современных систем производительностью в сотни и более TFLOPS для численного моделирования сложных физических процессов, сопряжено со значительными трудностями. Основой современного суперкомпьютерного парка являются вычислительные системы, оснащенные многоядерными процессорами и различными ускорителями (ПЛИС, GPU, Intel MIC и другими). Такие системы предъявляют особо жесткие требования к вычислительным алгоритмам, которые должны быть как можно проще и прозрачнее с логической точки зрения, но вместе с тем достаточно эффективны. Значительную проблему представляют вопросы создания программного обеспечения общего назначения. Отметим в этой связи необходимость разработки компиляторов, облегчающих перенос на новые архитектуры существующих прикладных программ; создания библиотек и средств обработки больших объемов сеточных данных, методов формирования геометрических моделей высокого качества, средств декомпозиции расчетных сеток; алгоритмов и библиотек балансировки загрузки процессоров; средств визуализации результатов крупномасштабных вычислительных экспериментов.

http://www.ospcon.ru/files/media/Jakobowskiy.pdf

Перспективы развития аппаратных технологий и их применение в суперкомпьютерах экзафлопсного уровня

Слепухин А. Ф. — «Т-Платформы»

http://www.ospcon.ru/files/media/Slepuhin.pdf

Суперкомпьютерные технологии в промышленности: опыт и актуальные задачи

Дерюгин Ю. Н., Костюков В. Е., Соловьев В. П., Шагалиев Р. М. — ИТМФ, РФЯЦ-ВНИИЭФ

Важнейшим фактором, позволяющим решить задачи обеспечения качества, повышения эффективности проектирования и производства, совершенствования тактико-технических характеристик выпускаемой продукции, является уровень внедрения современных суперкомпьютерных технологий (СКТ) в процесс производства перспективных образцов техники. В 2010–2012 годах в рамках проекта Комиссии при Президенте РФ по вопросам модернизации и технологического развития экономики России «Развитие суперкомпьютеров и грид-технологий», утвержденного Президентом РФ (далее Проект), решена задача по созданию базовых компонентов отечественных суперкомпьютерных технологий. Все принятые планы выполнены в срок и с надлежащим качеством, а по ряду направлений обеспечено опережение установленных сроков и превышение плановых показателей.

В результате выполнения Проекта созданы уникальные продукты, не имеющие отечественных аналогов, — в частности, разработаны пакеты программ для имитационного моделирования на суперкомпьютерах. Эта полностью отечественная разработка доступна всем российским предприятиям и организациям. Пакеты программ предназначены для массового применения на предприятиях ОПК и стратегических отраслей промышленности. По функциональному наполнению пакеты программ для имитационного моделирования позволяют решать почти 70% классов задач предприятий. В настоящее время пакеты программ прошли верификацию, валидацию и развернуты на более чем 200 рабочих местах на 22 предприятиях.

С использованием отечественных пакетов программ созданы пилотные версии виртуальных моделей перспективных образцов изделий авиастроения, атомной энергетики, автомобилестроения и ракетно-космической отрасли. Пилотные версии виртуальных моделей введены в опытную эксплуатацию на ряде ведущих предприятий (ОАО «Компания "Сухой"», ФКП «НИЦ РКП», ОАО «КБХА», ФГУП «ГНПРКЦ ЦСКБ ПРОГРЕСС», ОАО «СПбАЭП», ОАО «ОКБМ Африкантов», ОАО ОКБ «ГИДРОПРЕСС», ОАО «КАМАЗ»).

В ФГУП «РФЯЦ-ВНИИЭФ» создан и введен в эксплуатацию самый производительный в России суперкомпьютер — впервые в стране промышленным предприятиям выделены значительные вычислительные ресурсы для решения масштабных практических задач. Разработана технология удаленного доступа к этим ресурсам, позволяющая проводить расчеты в 50 предприятиях и организациях. Спроектирована новая для России линейка компактных суперкомпьютеров (КС-ЭВМ), ориентированных на массовое оснащение рабочих мест специалистов КБ. Организовано серийное производство КС-ЭВМ — на данный момент уже более 80 КС-ЭВМ поставлено на предприятия.

Достижение целей Проекта состоялось благодаря кооперации предприятий промышленности, науки и образования, а также ИТ-компаний (более 40 организаций) с госкорпорацией «Росатом» в роли государственного заказчика, а ФГУП «РФЯЦ-ВНИИЭФ» — головного исполнителя. В результате реализации Проекта создана основа для промышленного внедрения суперкомпьютеров — ключевого компонента новой технологии проектирования и разработки продукции на отечественных предприятиях оборонных и гражданских отраслей, а также для преодоления отставания России от Запада и устранения зависимости от зарубежных разработок.

Крайне актуальными на ближнесрочный период (2014–2018 годы) являются следующие задачи:

- промышленное внедрение отечественных суперкомпьютерных технологий в технологический цикл проектирования и производства изделий стратегических отраслей (ОПК, авиастроение, атомная энергетика, автомобилестроение, ракетно-космическая отрасль);
- расширение области применения отечественных суперкомпьютерных технологий для решения задач нефтегазовой отрасли;
- доведение созданных отечественных разработок до уровня продукции, способной конкурировать на рынке высоких технологий.

http://www.ospcon.ru/files/media/Deryugin.pdf

Путь к Exascale — опыт РСК создания масштабируемых суперкомпьютеров

Московский А. А. — «РСК Технологии»

Сегодня критически важными технологиями для создания масштабируемых суперкомпьютеров являются: энергоэффективность, большая вычислительная плотность, высокая отказоустойчивость и масштабируемая производительность. В докладе представлен опыт специалистов РСК, разработавших и внедривших в целом ряде российских организаций (МСЦ РАН, ЮУрГУ, МФТИ, Росгидромет и др.) передовые решения на базе инновационной архитектуры «РСК Торнадо» прямого жидкостного охлаждения стандартных и массово доступных электронных компонентов. Данная архитектура позволяет достигать рекордной на сегодняшний день энергоэффективности PUE = 1,057 за счет оптимизированного охлаждения и интеллектуального управления энергопитанием с расширенными возможностями мониторинга. Решениям от РСК принадлежит мировой рекорд по вычислительной плотности для архитектуры x86, составляющий сегодня 211 TFLOPS для стандартной стойки 42U. Уже сейчас решения и реальный опыт в эксплуатации систем РСК позволяют говорить о достижимости производительности экзаскалярного уровня и возможности создания систем такого класса в недалеком будущем. Для обеспечения высокой отказоустойчивости в РСК используются лучшие в мире электронные компоненты, включая стандартные серверные платы Intel, старшие модели процессоров Intel Xeon E5-2697 v2 и сопроцессоров Intel Xeon Phi 7120X, диски Intel SSD, а также передовые технологии повышения надежности работы оборудования. В докладе проанализировано влияние понижения рабочей температуры на сокращение количества ошибок и сбоев в работе оборудования. В контексте достижения высокомасштабируемой производительности вычислительные узлы на базе архитектуры «РСК Торнадо» позволяют получить максимальную отдачу при минимальном количестве нагруженных ядер — например, за счет возможности постоянной работы встроенной в серверные процессоры Intel технологии Turbo Boost, обеспечивающей за счет эффективного жидкостного охлаждения дополнительный прирост производительности 300 МГц на каждом ядре.

Высокая пропускная способность в системах РСК достигается за счет использования лучших в индустрии коммуникационных интерфейсов (например, Infiniband FDR), а гибкость решения обеспечивается благодаря предоставлению пользователям различных опций подключения внешних карт расширения (56+100 Гбит/узел Infiniband) в комплексе с выделенными сетями мониторинга и другими инновационными функциональными возможностями.

http://www.ospcon.ru/files/media/Shmelev.pdf

Высокопроизводительные микропроцессы выполнения однопоточных приложений

Ким А. К., Волконский В. Ю., Груздов Ф. А., Нейман-заде М. И., Семенихин С. В., Слесарев М. В. — ИНЭУМ им. И. С. Брука

Развитие ИТ в значительной степени определяется темпами развития таких ключевых областей, как микроэлектроника и программное обеспечение, характеризуемых экспоненциальным ростом числа транзисторов в микропроцессорах и экспоненциальным ростом объема и сложности программ. Экспоненциальные рост числа транзисторов в микропроцессорах и ограничения по мощности требуют создания новых параллельных архитектур, однако требования совместимости с уже существующим программным обеспечением, распространяемым в виде двоичных кодов для наиболее популярных аппаратных платформ, сдерживают развитие архитектуры процессорного ядра. Как следствие, разработчиками микропроцессоров используется стремительно растущий аппаратный параллелизм для увеличения числа процессорных ядер и расширения векторных регистров.

Как показали исследования [1], практически все программы обладают огромным потенциалом параллелизма на уровне операций — от нескольких десятков до нескольких тысяч операций за такт. Этот вид параллелизма наиболее универсален, он может быть эффективно поддержан в аппаратуре и автоматически обнаружен (с помощью компиляторов) в существующих программах. Векторный параллелизм (операции над упакованными данными) также поддается аппаратно-программной оптимизации, но имеет ограниченное применение в программах. Параллелизм потоков управления значительно труднее поддается программной автоматизации и зачастую требует существенных усилий программистов для явного распараллеливания программ. Таким образом, использование параллелизма на уровне операций — важнейший способ повышения производительности процессорного ядра и, как следствие, увеличения производительности многоядерных систем в целом за счет ускорения вычислений на участках, не поддающихся другим видам распараллеливания.

Быстрый рост объема и сложности ПО характеризуется снижением его общей надежности из-за большого количества ошибок и уязвимостей. Например, размер дистрибутива Linux Debian с 2001 года вырос более чем в 9 раз по числу входящих в него пакетов и почти в 9 раз по размеру исходного кода, а надежность составляет 1–5 ошибок на 1 тыс. строк кода. Последнее десятилетие характеризуется стремительным ростом киберпреступности и кибератак, что стало прямым следствием ошибок в программах, а также использованием аппаратных архитектур, не имеющих средств повышения надежности программ. Таким образом, вопросы безопасности и надежности программного обеспечения выходят в ИТ на первый план.

Решение перечисленных проблем возможно на базе микропроцессоров с архитектурой «Эльбрус» [2], которые обеспечивают высокую производительность процессорного ядра за счет параллельного исполнения операций при очень экономном энергопотреблении, полную и эффективную двоичную совместимость с платформой x86, x86-64, а также предоставляют средства надежного программирования на базе аппаратно поддерживаемых защищенных вычислений. Благодаря этому микропроцессоры с архитектурой «Эльбрус» могут быть рассчитаны на широкий спектр применений.

Ключевой идеей архитектуры «Эльбрус» является явное указание процессорному ядру параллельно исполняемых инструкций, при этом анализ независимости и планирование инструкций выполняет компилятор. С одной стороны, это позволяет отказаться от сложной и энергоемкой аппаратуры обеспечения внеочередного исполнения инструкций, используемой во всех современных универсальных микропроцессорах, а с другой, делает осмысленными расширение парка исполнительных устройств и повышение предельной производительности на такт до уровней, превосходящих возможности альтернативных решений.

Текущая реализация архитектуры «Эльбрус» позволяет запускать на исполнение с помощью широкой команды на ациклических участках кода до 16 операций, а при выполнении циклов — до 23 операций. Для снижения потерь при обращении к данным в памяти реализован аппаратно-программный механизм асинхронной предварительной подкачки в специальные буферы. Подготовки переходов обеспечивают возможность эффективно их осуществлять без использования предсказателя. Аппаратура поддерживает средства программно-управляемого переупорядочивания операций и одновременного исполнения нескольких условных ветвей программы.

Вместе с параллелизмом на уровне операций в архитектуре «Эльбрус» реализован векторный параллелизм за счет операций над упакованными данными, а также параллелизм потоков управления на базе многоядерности и многопроцессорности на общей памяти. Все эти механизмы эффективно поддержаны оптимизирующим компилятором [3], что позволяет автоматически получать коды, использующие мощное процессорное ядро и другие параллельные возможности архитектуры.

В архитектуре Эльбрус аппаратно реализована технология защищенного исполнения программ [4], позволяющая на порядок ускорить процесс поиска ошибок и за счет этого создавать более надежные программы, в которых контролируются все обращения в память для предотвращения выхода за разрешенный диапазон адресов, блокируется некорректная работа с указателями на локальные (временные) объекты и диагностируется использование неинициализированных данных. Одним из приложений данной технологии является создание особо защищенных систем, практически неуязвимых для кибератак.

Одной из важных проблем при переходе на новую архитектуру является перенос существующего ПО — например, для популярной архитектуры x86 (x86-64) накоплен огромный объем программного обеспечения, распространяемого в виде двоичных кодов, и для их эффективного исполнения на архитектуре «Эльбрус» в ней имеются специальные аппаратные средства обеспечения совместимости. Такие средства не требуют повторения системы команд исходной платформы Intel (это невозможно из-за лицензионных ограничений). Аппаратные средства обеспечивают эффективность и надежность режима совместимости, который реализуется с помощью программного метода многоуровневой динамической двоичной трансляции кодов Ix86 в коды «Эльбрус» [5]. Этот подход позволяет достигать производительности, сопоставимой с современными микропроцессорами Intel.

С момента появления первого микропроцессора линии «Эльбрус» в 2007 году создано четыре поколения этих изделий, используемых в различных решениях от встраиваемых систем до серверов. Процессор третьего поколения — система на кристалле «Эльбрус-2С+» — изготовлен с использованием технологии 90 нм, содержит два ядра архитектуры «Эльбрус» и работает на частоте 500 МГц. Результаты замеров производительности, проведенные ко-

миссией Минпромторга, подтверждают высокую логическую скорость и сопоставимость с процессорами Intel Core2 3 ГГц.

Сегодня изготовлен по технологическим нормам 65 нм и готовится к государственным испытаниям микропроцессор четвертого поколения с архитектурой «Эльбрус-4С», имеющий 4 ядра, увеличенную кэш-память и 3 независимых канала работы с памятью. В микропроцессор включены средства повышения эффективности — как при исполнении откомпилированных программ, так и в кодах x86-64. Ожидается, что будет достигнута рабочая частота 1 ГГц, а процессор может представлять практический интерес для производителей суперкомпьютеров.

Дальнейшее развитие линейки «Эльбрус» связано с увеличением количества операций в процессорном ядре, увеличением числа ядер, архитектурными улучшениями и с переходом на более «тонкие» нормы производства. Ведется разработка пятого поколения микропроцессоров с архитектурой «Эльбрус» 8-ядерного процессора «Эльбрус-8С» (28 нм), вычислительная мощность которого составит 250 GFLOPS, а выпуск ожидается в 2015 году. В программу развития этой линии микропроцессоров также включены разработка и выпуск следующих поколений процессоров «Эльбрус», достигающих вычислительной мощности более 1 TFLOPS на кристалл.

http://www.ospcon.ru/files/media/Volkonskiy.pdf

Литература

1. Postiff M.A., Greene D.A., Tyson G.S., Mudge T.N. The Limits of Instruction Level Parallelism in SPEC95 Application // INTERACT-3 at ASPLOS-VIII, 1998.

2. Ким А.К. Российские универсальные микропроцессоры и вычислительные комплексы высокой производительности: результаты и взгляд в будущее. // Вопросы радиоэлектроники. Сер. ЭВТ. 2012. Вып. 3. С. 5-13.

3. Волконский В.Ю., Брегер А.В., Бучнев А.Ю., Грабежной А.В., Ермолицкий А.В., Муханов Л.Е., Нейман-заде М.И., Степанов П.А., Четверина О.А. Методы распараллеливания программ в оптимизирующем компиляторе // Вопросы радиоэлектроники. Сер. ЭВТ. 2012. Вып. 3. С. 63-88.

4. Ким А.К., Волконский В.Ю., Груздов Ф.А., Сахин Ю.Х., Семенихин С.В. Защищенное исполнение программ на базе аппаратной и системной поддержки архитектуры «Эльбрус» // V Международная научно-практическая конференция «Современные информационные технологии и ИТ-образование». М.: Изд-во МГУ, 2010. С. 22-39.

5. Воронов Н.В., Гимпельсон В.Д., Маслов М.В., Рыбаков А.А., Сюсюкалов Н.С. Система динамической двоичной трансляции x86->«Эльбрус» // Вопросы радиоэлектроники. Сер. ЭВТ. 2012. Вып. 3. С. 89-107.

Семь дней из жизни суперкомпьютера

Воеводин В. В., Антонов А. С., Жуматий С. А., Никитенко Д. А., Стефанов К. С. — НИВЦ МГУ

Сегодня активно обсуждается образ экзафлопсного компьютера, его основные параметры, базовая архитектура, делаются оценки числа ядер на кристалле и числа ядер на узле, указывается необходимый темп его работы с памятью, даются оценки размеров оперативной памяти на узел, активно дискутируются изменения стека программного обеспечения и многие другие компоненты программно-аппаратной среды. Вместе с тем успешная работа будущих суперкомпьютерных центров будет определяться не только этими характеристиками — появление систем экзафлопсной производительности на порядок увеличит число пользователей, поддержка эффективной работы которых потребует исключительно серьезного внимания. Значительно вырастет число компонентов установленного программного обеспечения, что потребует постоянного мониторинга их корректной работы [1]. Увеличится число компонентов самого суперкомпьютера, что потребует пересмотреть сложившуюся технологию его техподдержки, обслуживания и сопровождения. Уже сейчас возникает множество вопросов, на первый взгляд выходящих за предмет научного исследования, но требующих обязательного решения, иначе будущие суперкомпьютерные центры и системы заведомо будут обречены на неэффективную работу. В докладе сообщается о подходах к решению и особенностях вопросов, возникающих в ежедневной практике сопровождения суперкомпьютерного комплекса Московского университета.

http://www.ospcon.ru/files/media/Voevodin.pdf

Литература

1. Воеводин Владимир, Воеводин Вадим. Спасительная локальность суперкомпьютеров // Открытые системы. 2013. № 9. С. 12-15.

Секция. Экзафлопсные архитектуры

Архитектура потоковой вычислительной системы экзафлопсного уровня производительности

Стемпковский А. Л., Левченко Н. Н., Окунев А. С., Климов А. В. — ИППМ РАН

Одной из центральных проблем, которую приходится решать создателям масштабируемых параллельных программ, является эффективная доставка данных к вычислителям. В парадигме фон Неймана основным инструментом решения этой проблемы является кэш-память, которая создает у программиста иллюзию того, что проблемы нет, однако с ростом объемов данных, и особенно в связи с переходом на мультипроцессорные системы, она сама стала создавать новые проблемы, такие как:

- плохая предсказуемость времени доступа;
- необходимость введения сложных и вносящих замедление механизмов поддержки когерентности кэш-памятей в многопроцессорных системах;
- неоптимальность универсальных автоматических алгоритмов вытеснения и подкачки, порождающая потребность в ручном управлении перемещением данных между уровнями;
- сложность работы с несколькими уровнями кэш-памяти, за которыми следует основная память, затем дисковая и т. д.

При использовании графических ускорителей проблема еще больше обостряется — в них появляется нескольких новых видов памяти: текстурная (память констант), глобальная, разделяемая. В результате на программиста наваливается груз забот об оптимизации перемещений между этими видами памятей. И фоннеймановская модель вычислений уже не помогает, а только мешает, поскольку запрос на элемент данных в ней возникает только тогда, когда этот элемент уже нужен для вычислений, — и создается проблема борьбы с задержками. Нужна модель вычислений, которой была бы свойственна упреждающая подача данных в вычислитель. Предлагаемая в докладе архитектура реализует потоковую модель вычислений в парадигме раздачи (ППР), которая и обладает указанным свойством.

В ППР значительно упрощается написание программ, оптимальных с точки зрения коммуникационного трафика и трафика по обращениям к памяти. Работа в парадигме «раздачи» заключается в следующем. Обычно всякий элемент данных, возникающий в процессе вычислений, имеет одного производителя и одного или нескольких потребителей — всякое вычислительное действие является потребителем по одним данным и производителем по другим. В парадигме «сбора», свойственной фоннеймановским языкам, производитель помещает результат в определенное место памяти, и в дальнейшем потребители запрашивают (собирают) нужные им данные непосредственно перед использованием. В парадигме «раздачи», наоборот, производитель знает, кому потребу-

ется его результат, и передает его в форме сообщения, указав адрес потребителя. Потребитель может и не знать, откуда к нему придет аргумент, используя для доступа к нему локальное имя. Для организации доставки данных у всех вычислительных действий должны быть уникальные адреса. Обычно это вектора индексов, имеющие содержательный смысл, связанный с задачей.

Наличие адресации у всех вычислительных действий дополнительно создает возможность единообразно управлять распределением вычислений по пространству и времени, задавая привязку виртуальных адресов к физическому пространству или времени. Привязка задается путем указания функции распределения, аргументом которой является сам адрес, а результатом — координаты в пространстве (номер процессора) или времени (номер этапа). Имея эту информацию, аппаратура системы в состоянии автоматически обеспечить оптимальную доставку данных в нужное место в нужное время.

Аппарат функций распределения — простое и гибкое средство, позволяющее в широких пределах варьировать порядок вычислений, их положение в пространстве и, как следствие, управлять параллелизмом, минимизируя перемещения данных как между процессорами, так и между уровнями памяти. При этом важно, чтобы само описание алгоритма не навязывало никакого порядка вычислений, кроме порядка, индуцируемого информационными зависимостями. Именно так обстоит дело в модели вычислений ППР, где возможность произвести то или иное вычисление определяется только готовностью аргументов. При этом возможны выполнение, отладка и тестирование программ еще до задания функций распределения, которые повлияют только на итоговую эффективность и масштабируемость.

http://www.ospcon.ru/files/media/Klimov.pdf

Графические процессоры в суперкомпьютерных системах

Корнеев В. В., Павлухин П. В., Шевченко И. В. — НИИ «Квант»

Высокая удельная производительность на единицу оборудования графических процессоров (GPU) обусловлена использованием синхронных тредов, оформленных как подмножества (варпы) тредов. При невозможности продолжения вычислений текущего варпа из-за ожидания завершения обращения к памяти происходит быстрое переключение на исполнение другого варпа, треды которого готовы к исполнению. Для эффективной работы в ускорителях организуется два основных конвейера команд: вычислительный и управляющий доступом в память. Графические процессоры имеют многоуровневую организацию памяти, поэтому для получения реальной высокой производительности требуется дополнительное изучение деталей совмещения работы вычислительных и управляющих конвейеров.

Создание высокоэффективных приложений для GPU требует глубокого знания особенностей их микроархитектуры, но многие важные характеристики (максимальное число поддерживаемых («на лету») одновременных обращений в память на варп, латентность кэшей L1, L2 и др.) часто в официальной документации производителей не указываются. Написание тестов для их выяснения сопряжено с рядом трудностей — производители обычно не предоставляют низкоуровневых средств программирования для GPU. Однако в ряде случа-

ев может быть реализован ассемблер для ускорителей поколения GPU с поддержкой всех основных инструкций. С помощью этого языка было проведено тестирование и получены значения латентностей при доступе к различным видам памяти; показана важность использования не только параллелизма на уровне тредов, но и на уровне инструкций; исследована работа конвейеров GPU.

На некоторых классах задач с использованием ускорителей удается добиться прироста производительности на один-два порядка по сравнению с CPU, однако во многих из них сложно получить хорошую масштабируемость на многопроцессорных системах, поскольку значительный вклад в общее время работы программы начинает вносить обмен данными между микропроцессорами. Для улучшения производительности на таких системах возможно применение различных техник совмещения счета на GPU с двусторонней передачей данных CPU — GPU и межузловыми обменами. В работе проведено исследование факторов, влияющих на производительность взаимодействия GPU на примере CUDA и MPI.

GPU — будущее гибридных вычислительных систем

Джораев А. Р. — NVIDIA

http://www.ospcon.ru/files/media/Dzhorayev.pdf

Решения Intel: путь к Exascale

Местер Н. С. — Intel

http://www.ospcon.ru/files/media/Mester2.pdf

Особенности применения графических процессоров для ускорения параллельных вычислений

Щекалев А. С. — Yandex

Машинное обучение играет ключевую роль во многих областях науки и промышленности, имеет большое значение для статистики, анализа данных и искусственного интеллекта, а также находит широкое применение в информационном поиске. Поисковая система «Яндекс» уже давно использует машинное обучение для решения важной задачи ранжирования, которая заключается в распределении документов, полученных на конкретный запрос, в порядке, максимально близком к идеальному ранжированию, сделанному на основе экспертных оценок. Документы представлены набором факторов, описывающих их различные свойства (содержание слов запроса в тексте, количество ссылок, возраст документа и т. д.), а мера близости определяется заданной функцией потерь.

Алгоритмы обучения ранжированию можно разделить на три группы в зависимости от способа представления данных: поточечные, попарные и позапросные. В поточечном случае задача ранжирования может быть представлена в виде обычной задачи регрессии, которая заключается в предсказании оценки для заданной пары запрос-документ. Попарный подход сводится к построению классификатора, который определяет лучший документ в паре, полученной для заданного запроса. Позапросный способ оптимизирует среднее значение функции потерь по всем запросам, доступным для обучения.

Для решения регрессионной задачи имеется много различных методов, отличающихся вычислительной сложностью. Существенное развитие сегодня получили методы, основанные на построении ансамблей, — алгоритмы бустинга и бэгинга. Идея бустинга заключается в построении модели в виде линейной комбинации простых функций. Это итеративный процесс, который начинается с некоторого начального предположения и последовательно добавляет новые функции. Благодаря такому подходу, становится возможно менять качество и эффективность полученных моделей на машинное время их расчета. Таким образом, проблема производительности подбора перестала быть второстепенной и вышла на первый план наряду со сбором данных и разработкой математической модели.

Конкретная реализация простых функций может зависеть от решаемой задачи. Многие исследования показали, что выбор деревьев решений в качестве слабых моделей позволяет добиться наилучших результатов в задаче ранжирования. Дерево разделяет векторное пространство факторов на несколько непересекающихся областей и для каждой из них предсказывает постоянное значение. Дерево определяется конфигурацией его листьев и порядком

подсчета оптимального решения в рамках этих листьев. Например, в качестве конфигурации может выступать набор условий, ограничивающих значения фактора, а оптимальное решение — определяться средним значением целевой функции по всем элементам, которые вошли в лист. Тот факт, что области не пересекаются, позволяет распараллелить процесс построения дерева решений.

В современном мире для параллельных и распределенных вычислений уже давно используются суперкомпьютеры с большим количеством центральных процессоров, а с другой стороны, развитие компьютерной техники привело к появлению специализированных вычислительных устройств, таких как графические ускорители, обеспечивающих большую теоретическую производительность.

С аппаратной точки зрения графические процессоры состоят из масштабируемого набора многозадачных потоковых мультипроцессоров, позволяющих запускать сотни вычислительных потоков одновременно, но и не обязательно обеспечивающих при этом высокую эффективность. Потоки запускаются группами по 32 элемента, а переключение между ними осуществляется мгновенно, так как контекст выполнения содержится на мультипроцессоре.

Сегодня на рынке имеется ряд программных технологий, позволяющих воспользоваться ресурсами GPU для выполнения общих вычислений: шейдеры, CUDA и OpenCL. В силу сво-

ей специализации шейдеры накладывают серьезные ограничения на алгоритм вычислений. OpenCL интересен тем, что является универсальным стандартом параллельных вычислений, не зависящим от конкретного аппаратного обеспечения, однако в решениях Nvidia он поддерживается недостаточно полно, поэтому для эффективных коммерческих реализаций пока предпочтителен вариант CUDA.

Обучение ранжированию в поисковой системе «Яндекс» реализовано с помощью алгоритма MatrixNet, который, в свою очередь, основан на градиентном бустинге деревьев решений. Использование градиента позволяет упростить задачу оптимизации с ранжирующей функцией в виде линейной комбинации. В качестве множества простых функций MatrixNet использует множество «забывчивых» деревьев решений, включающих 6 разбиений и 64 листа. Алгоритм не рассматривает все возможные деления — это слишком трудоемко, а вместо этого для каждого фактора выбираются 32 значения, которые выступают в качестве возможного деления пространства, — именно они рассматриваются при построении дерева.

Учитывая ограничение на количество делений одного фактора, в основу составления дерева можно положить вычисление 32-разрядных гистограмм распределения документов для каждого фактора. Значения в ячейках гистограмм будут использованы для оценки разбиения. Вычисление гистограммы очень хорошо подходит для архитектуры графического процессора и позволяет задействовать наиболее производительные механизмы.

Для оценки производительности GPU реализации было произведено сравнение скорости выполнения одной итерации алгоритма MatrixNet на специальном стенде в различных режимах в зависимости от размера данных. Запуски производились с использованием процессора Intel Xeon E5-2650 для 8 и 16 вычислительных потоков, в качестве GPU служил ускоритель Tesla C2075. Для обучения применялись актуальные данные, на которых обучаются текущие формулы ранжирования поисковой машины «Яндекс» и которые описываются более чем 700 факторами. Увеличение количества потоков центрального процессора вдвое приводит к соответствующему росту производительности, однако применение только одного графического ускорителя позволяет ускорить процесс более чем в 20 раз [1].

Графические процессоры очень эффективны при обучении ранжированию, например для построения деревьев решений. В первую очередь это связано с тем, что алгоритм построения деревьев допускает высокую степень параллелизма и требует значительных вычислений. Компания «Яндекс» широко применяет графические процессоры для оптимизации вычислительных процессов: в частности, MatrixNet — основной метод машинного обучения — полностью реализован на GPU. Кроме того, графические процессоры помогают ускорить проверку новых поисковых факторов, существенно экономя аппаратные ресурсы, — например, для проверки одного фактора используется 500 однопроцессорных серверов, однако эту задачу за то же время решают шесть машин, оснащенных четырьмя графическими ускорителями.

http://www.ospcon.ru/files/media/Shekalew.pdf

Литература

1. Кураленок И., Щекалев А. GPU в задачах машинного обучения // Открытые системы. 2013. № 8. С. 44-46

На пути к созданию отечественного суперкомпьютера субэкзафлопсной производительности

Михеев В. А., Изгалин С. П., Леонова А. Е., Фролов А. С., Симонов А. С., Слуцкин А. И. — НИЦЭВТ

На пути развития суперкомпьютеров во второй декаде нового тысячелетия возник ряд проблем, связанных с преодолением множества технологических барьеров, самым заметным из которых является достижение экзафлопсной производительности в условиях ограничений на размер вычислительной системы, энергопотребление, стоимость и других факторов, среди которых есть ограничения экономического и политического характера. Именно последние ограничения заставляют Китай, Японию и Евросоюз вкладывать огромные средства в развитие электронной компонентной базы, создавать и поддерживать собственные технологии разработки микропроцессоров, высокоскоростных коммуникационных сетей, системного и прикладного программного обеспечения, а самое главное — создавать и развивать собственные коллективы, способные решать самые сложные задачи в области построения и использования суперкомпьютеров.

В России состояние микроэлектроники трудно сегодня назвать инновационным, очевидно, что все проблемы в один момент не решаются, однако в последнее время были сделаны некоторые правильные шаги. Заметен прогресс в области разработки отечественных микропроцессоров (МЦСТ, НИИСИ РАН), высокоскоростных коммуникационных сетей (НИЦЭВТ, НИИ «Квант», РФЯЦ ВНИИЭФ и др.), идет внедрение новых технологических процессов на российских полупроводниковых фабриках ОАО «НИИМЭ и Микрон» и «Ангстрем».

В ОАО «НИЦЭВТ» с 2005 года ведется проект по созданию отечественной высокоскоростной коммуникационной сети «Ангара» с топологией 4D-тор, основная задача которого — создать предпосылки для полного или частичного импортозамещения в российских вычислительных центрах, ориентированных на решение важнейших для государства инженерных и научно-технических задач.

Итогом стала появившаяся в 2013 году СБИС ЕС8430, содержащая 180 млн транзисторов, 112 высокоскоростных приемопередатчиков, размещенных по трем сторонам кристалла. В таблице приведены основные характеристики сети «Ангара» на базе СБИС, макета сети «Ангара» на базе ПЛИС, сетей Mellanox InfiniBand FDR 4x, IBM Blue Gene/Q, Cray XK7 (Cray Gemini).

Сеть «Ангара» поддерживает топологию «кольцо», 2D-, 3D- и 4D-тор, детерминированную и адаптивную передачу пакетов, односторонние операции с удаленной памятью (запись, чтение, атомарные операции). Аппаратно поддерживаются механизмы синхронизации, записи в кольцевые буфера, коллективные операции.

На канальном и сетевом уровнях реализованы отказоустойчивые протоколы передачи данных. Основным режимом программирования для сети «Ангара» является совместное использование MPI, OpenMP и SHMEM. Также поддерживаются средства параллельного программирования GASNet, UPC, ARMCI, Charm++.

Сравнительные характеристики сети «Ангара»

Характеристика	«Ангара М3» (ПЛИС)	«Ангара» (СБИС)	InfiniBand FDR 4x	IBM BG/Q	Cray XK7
Топология сети	2D-тор	4D-тор	fat tree	5D-тор	3D-тор
Пропускная способность с процессором, Гбайт/с	2	8	8	~20	9,6
Пропускная способность соединения, Гбайт/с	0,625	7,5	6,8	2	9,375
Агрегатная пропускная способность, Гбайт/с	5	120	н/д	40	186
Задержка между соседними узлами, мкс	2,5	1,0	1,0	< 1,0	1,4

СБИС ЕС8430 выпущена на фабрике TSMC с использованием технологии 65 нм. Размер кристалла — 13,0×10,5 мм в корпусе flip-chip BGA с 1521 выводом в виде массива 39×39 контактов с шагом 1 мм, размер подложки — 40×40 мм. СБИС работает на частоте 250/500 МГц и потребляет 36 Вт. В рамках поддерживаемой топологии сети «четырехмерный тор» каждый сетевой узел может иметь до 8 соединений с соседними узлами, пропускная способность каждого соединения — 75 Гбит/с (12 линий по 6,25 Гбит/с, кодирование 8b10b). Взаимодействие с вычислительным узлом осуществляется через интерфейс PCI Express Gen2 x16 (80 Гбит/с, кодирование 8b10b).

СБИС будет использована в сетевых адаптерах сети «Ангара» — платах расширения PCI Express для кластерных систем с коммерчески доступными процессорами, а также в составе разрабатываемой в НИЦЭВТ в рамках проекта «Ангара» вычислительной платформы.

Сам факт успешного выпуска СБИС адаптера/маршрутизатора высокоскоростной коммуникационной сети, полностью разработанной отечественным коллективом, — это уникальное событие, реальный шаг на пути к созданию отечественного суперкомпьютера субэкзафлопсной производительности, значительно сокращающий отставание страны от ведущих мировых держав в области высокопроизводительных вычислений.

http://www.ospcon.ru/files/media/Simonov.pdf

Системное ПО для суперкомпьютеров: проблемы, решения, перспективы

Аветисян А. И., Кошелев В. К., Кудрявцев А. О. — ИСП РАН

Все современные суперкомпьютеры — системы с распределенной памятью, содержащие сотни тысяч ядер традиционной архитектуры и/или миллионы ядер ускорителей вычислений. Масштабируемость таких систем обеспечивается как на уровне аппаратуры, так и

на уровне системного ПО (ОС, среды времени выполнения, средств разработки). В докладе дается обзор текущего состояния, проблем и перспектив развития системного ПО перспективных экзафлопсных систем.

http://www.ospcon.ru/files/media/Arut.pdf

Элементы программирования параллельных экзафлопсных систем

Штейнберг Б. Я., Гервич Л. Р., Юрушкин М. В. — ЮФУ

Данная работа посвящена проблемам программирования экзафлопсных вычислительных систем, появление которых ожидается в ближайшие десятилетия. Известно, что для различных задач для достижения хорошей эффективности требуются процессоры различных архитектур [1], а архитектуры суперкомпьютеров в еще большей степени, чем входящие в их состав процессоры, влияют на эффективность решаемых задач. Будем полагать, что создаваемый экзафлопсный суперкомпьютер будет подобен существующим ныне, только больше, а микросхемы, входящие в его состав, будут содержать больше элементов. Для такого суперкомпьютера будет рассматриваться проблема создания в разумные сроки программного обеспечения, близкого по производительности и к рекордному.

В течение нескольких десятилетий с каждым годом уменьшается отношение времени выполнения арифметической операции ко времени считывания аргументов этой операции из памяти [2]. Пятьдесят лет назад, когда проектировались первые параллельные суперкомпьютеры, умножение вещественных чисел было очень дорогостоящей операцией, а обращения к памяти требовали незначительного времени по сравнению с умножением. На сегодняшний день — наоборот, обращение к памяти на порядки дольше, чем выполнение арифметических операций, что привело к необходимости использования модулей вспомогательной памяти с быстрым доступом, которые технологически удается сделать небольшими по объему. Это может быть как кэш-память, управляемая процессором, так и локальная программируемая память (например, у процессора IBM Cell). Так возникла иерархия памяти, которую необходимо учитывать при разработке быстрых программ.

Доминирование сложности по обращениям к памяти объясняет неэффективность известного алгоритма перемножения матриц Штрассена по сравнению со стандартным алгоритмом, хотя он имеет лучшую вычислительную сложность по арифметическим операциям.

Элементы программирования высокопроизводительных вычислительных систем определяются их архитектурой. Даже если речь идет о высокоуровневом программировании, то оно должно быть ориентированным на архитектуру вычислительной системы. В работе [1] показано, что требования к архитектуре компьютера определяются задачами, которые предполагается на этом компьютере решать.

Различные композиции элементов памяти и вычислительных устройств схем дают многообразие сочетаний модулей памяти и вычислительных устройств, которое моделирует широкое множество современных компьютерных систем. Например, у многоядерного процессора есть общая кэш-память, и у каждого ядра есть еще своя кэш-память более высокого уровня. Разумеется, описание вычислитель-

ных архитектур не сводится к схемам соединений элементов памяти и вычислительных устройств, но эти схемы в значительной степени определяют быстродействие различных алгоритмов.

Во многих современных суперкомпьютерах можно проследить иерархию вычислительных устройств, и каждое такое устройство соответствует элементу иерархии памяти. Приведем один из распространенных примеров такой иерархии:

- вычислительное ядро со своей локальной (кэш-) памятью;
- многоядерный процессор: множество ядер на кристалле с общей для них памятью, расположенной на этом же кристалле;
- вычислительный узел: несколько кристаллов с одной общей оперативной памятью;
- кластер: несколько вычислительных узлов, соединенных коммуникационной сетью (возможно, с общей внешней памятью).

Во всех этих случаях схема соединений модулей памяти и вычислительных устройств может быть описана комбинацией элементарных схем.

Иерархия вычислительных устройств предполагает разбиение задачи на иерархию подзадач — на вычислительном устройстве каждого уровня архитектуры решается подзадача соответствующего уровня иерархии подзадач, причем разбиение задач на подзадачи необходимо даже для простейшей иерархической структуры памяти. Адекватное разбиение задачи на иерархию подзадач определяет эффективность вычислений. Следует подчеркнуть, что компиляторы делают такое разбиение автоматически и, как правило, без оптимизации.

Рассмотрим программу перемножения матриц:

```
do i = 1, n
do j = 1, n
do k = 1, n
X(i,j) = X(i,j)+A(i,k)*B(k,j).
```

Если матрица не помещается в кэш-память, то в кэш попадут строка матрицы A и столбец B, которые необходимы для вычисления самого вложенного цикла. В этом случае компилятор разбивает задачу перемножения матриц на подзадачи, состоящие в вычислении скалярных произведений. Тогда на каждую арифметическую операцию будет приходиться одно обращение к памяти (чтение). Если чтение чисел из оперативной памяти в 15 раз дольше перемножения чисел, то процессор будет загружен лишь на 1/15 своей мощности. Для повышения эффективности вычислений в этой задаче переходят к блочному произведению матриц. Основная цель разбиения задач на подзадачи состоит в том, чтобы данные каждой подзадачи помещались в памяти соответствующего в иерархии вычислительного устройства, а подзадач должно быть мало. Однако оба условия противоречат друг другу. Например, в качестве подзадачи можно взять выполнение одной операции — второе условие будет выполняться, но подзадач будет столько, сколько операций. Если в качестве подзадачи взять саму исходную задачу, то ее данные могут не поместиться в быстрой (кэш-) памяти. Баланс между этими условиями приводит к следующим принципам разбиения задачи на подзадачи:

- подзадачи должны формироваться так, чтобы с ограниченным (быстрой памятью) объемом данных можно было выполнять много операций;

- каждая подзадача должна быть максимально оптимизирована для своего уровня иерархии.

Применительно к перемножению матриц этот принцип ведет к разбиению матриц на блоки нескольких уровней и к соответствующему изменению кода. Следует отметить, что блоки могут быть и не квадратными [3].

Учет иерархии памяти для задачи перемножения матриц проходит достаточно легко, поскольку приводит к таким же подзадачам, как исходная, только меньшей размерности, но, например, для сортировки слиянием разбиение на подзадачи приводит к изменению алгоритма — вместо парного слияния возникает множественное, требующее использования дополнительной сложной структуры данных (например, 2–3 дерева).

Компиляторы имеют стандарты распределения данных, в частности, компилятор языка Фортран размещает матрицы в оперативной памяти по столбцам, а компиляторы языков Си и Паскаль — по строкам. Для минимизации кэш-промахов могут использоваться нестандартные размещения многомерных массивов — например, блочное размещение позволяет получить произведение матриц быстрее, чем у программ известных библиотек.

Для распределенной памяти в группе Оптимизирующей распараллеливающей системы [4] исследуются размещения с перекрытиями, суть которых в том, что, потратив (до 7%) память на дублирование некоторых данных, можно выиграть в быстродействии (несколько десятков процентов). Ведутся работы по автоматизации блочных распределений массивов в оперативной памяти и блочно-аффинных размещений массивов с перекрытиями в распределенной памяти.

Следует отметить, что рекордная производительность достигается непросто — например, рекордная программа перемножения матриц занимает порядка 1000 строк кода, среди которых много ассемблерных, учитывающих различные особенности архитектуры процессора, что примерно в 100 раз больше объема кода обычной программы. Программы с оптимизированным размещением данных в распределенной памяти также занимают существенно больший объем кода, чем обычные параллельные программы. Кроме увеличенного времени разработки, рекордные программы требуют очень высокой квалификации программистов. Сокращение времени разработки может быть достигнуто средствами автоматической оптимизации, которые на сегодняшний день развиты недостаточно.

Итак, видны следующие признаки эффективного ПО для суперкомпьютеров:

1. Описание иерархии подзадач, соответствующей иерархии архитектуры суперкомпьютера.

2. Использование структур данных, ориентированных на минимизацию обращений к модулям памяти, удаленным от вычислительного ядра.

3. Автоматизация разработки некоторых элементов ПО с помощью соответствующих новых инструментов.

http://www.ospcon.ru/files/media/Steinberg.pdf

Литература

1. Штейнберг Б.Я. Зависимость оптимального распределения площади кристалла процессора между памятью и вычислительными ядрами от алгоритма // Международная конференция «Параллельные вычисления и задачи управления» (PACO'2012) (Москва, 24-26 октября 2012 г.): труды. М.: Изд-во ИПУ РАН, 2012. С. 99-108.

2. Корнеев В.В. Проблемы программирования суперкомпьютеров на базе многоядерных мультитредовых кристаллов // Всероссийская суперкомпьютерная конференция «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность» (Новороссийск, 21-26 сентября 2009 г.): труды. М.: Изд-во МГУ, 2009.

3. Kazushige Goto, Robert A. van de Geijn. Anatomy of High-Performance Matrix Multiplication. ACM Trans. Math. Softw., Vol. 34, No. 3. (May 2008), pp. 1-25.

4. Гервич Л., Штейнберг Б., Юрушкин М. Программирование экзафлопсных систем // Открытые системы. 2013. № 08. С. 26-29.

Секция. Практика и кадры суперкомпьютерной индустрии

Символьные и квантовые вычисления для решения задачи выполнимости булевых формул

Гердт В. П. — ОИЯИ

Задача выполнимости булевых функций (формул), или задача SAT (сокращение от английского слова satisfiability), играет важную роль в современной теоретической и практической информатике. Согласно формулировке, для заданной булевой функции требуется установить, можно ли добиться ее истинности, предписывая определенное значение («истина» или «ложь») каждой из переменных. Интерес к задаче SAT в теоретической информатике обусловлен тем, что она NP-полная — ее эффективное (то есть с полиномиальным по числу переменных временем счета) решение будет означать эффективное решение многих вычислительно сложных задач переборного типа. Математически это будет означать, что P = NP, и дает тем самым решение еще одной из фундаментальных задач тысячелетия, а пока единственной решенной задачей этого класса является доказательство гипотезы Пуанкаре, найденное Г. Перельманом.

Практическая значимость задачи SAT обусловлена ее ролью во многих актуальных прикладных исследованиях, таких, в частности, как биоинформатика, управление удаленными космическими объектами, верификация программ и логического оборудования современных суперкомпьютеров. По данным компании Intel, до 70% затрат на разработку процессоров с новой микроархитектурой занимает их функциональная верификация. И здесь современные SAT-солверы (программы, реализующие алгоритмы переборного типа, с возвратом либо вероятностные) незаменимы. Тем не менее, ввиду быстрого роста числа логических элементов в современных микросхемах, необходим поиск новых методов решения задачи выполнимости.

В докладе представлены результаты применения методов символьных и квантовых вычислений к решению задачи SAT. Применение символьных вычислений к таким задачам основано на представлении частей булевой функции, заданной в конъюнктивной нормальной форме (КНФ), скобок КНФ-полиномов Жегалкина и приведении полученной системы полиномов к подходящей канонической форме. Одной из таких форм является результирующий полином Жегалкина, получаемый путем произведения скобочных многочленов и приведения подобных. В этом случае задача выполнима тогда и только тогда, когда полученный полином отличен от 1. Другой канонической символьной формой булевой функции, также позволяющей немедленно получить ответ для задачи SAT, является система полиномов Жегалкина специального вида, эквивалентная скобочной системе и получившая

название «базис Гребнера». Недавно разработан и программно реализован ряд алгоритмов построения этой второй канонической формы. Сейчас ведется работа по оптимизации этих алгоритмов и программ с целью сделать их конкурентоспособными среди лучших решателей SAT.

Квантовый подход к решению задачи SAT в ее оптимизационной формулировке (MAX 2-SAT) использует метод адиабатического квантового компьютинга, который является одним из наиболее многообещающих подходов в области квантовых вычислений и их приложений. В частности, именно на этот подход ориентированы технологии, используемые канадской фирмой D-Wave. Компьютерные эксперименты, выполненные недавно на основе 128-кубитного квантового процессора DW1 (Rainier) D-Wave, показывают его превосходство над AKMAXSAT — одним из лучших классических решателей задачи MAX 2-SAT.

http://www.ospcon.ru/files/media/Gerdt.pdf

Суперкомпьютеры для задач молекулярной динамики и квантовой механики молекул

Кузьминский М. Б. — ИОХ РАН

В докладе дан обзор применения систем суперкомпьютерного уровня в квантовой химии и молекулярной динамике и приведены примеры высокой практической актуальности таких задач. Показана возможность эффективного распараллеливания этих задач для определенных методов расчета и объектов исследования. Рассматриваются суперкомпьютеры уровня от MPP-систем, включая системы с собственными межсоединениями (IBM, Cray, Fujitsu), до кластеров стандартной архитектуры. Рассмотрены системы, в которых используются графические сопроцессоры (GPU), такие как NVIDIA Fermi и Kepler. Кратко рассмотрены и их возможные узкие места, например необходимость обмена данными хост- GPU по трактам PCI-E.

GPU Kepler лидирует сегодня по производительности выполнения операций с плавающей запятой — 1,3 TFLOPS. В представленной работе выполнены исследования GPU K20c и использованы внутренние тесты cuda-5.5. На тестах умножения матриц с применением библиотеки cuBLAS достигается производительность 1,2 TFLOPS, но тесты пропускной способности обменов данными хоста и GPU дают около 6,4 Гбайт/с в каждом направлении. Более ранние данные для C2050 на той же шине PCI-E v.2 x16 — около 6 Гбайт/с, что лишь немного ниже. В конечном счете эти величины лимитируются пропускной способностью шины (пиковое значение — 8 Гбайт/с).

Tesla C2050 и K20c с архитектурами Fermi и Kepler соответственно используются и в работах авторов доклада, создающих новый квантово-химический метод с аппроксимацией фокиана метода DFT, близкий к последнему по точности, но имеющий быстродействие на уровне полуэмпирических методов AM1, PM3 и др. Линейное масштабирование на гигантских молекулах в этом методе достигается за счет замены лимитирующей расчет диагонализации фокиана с асимптотической производительностью на метод типа PDM с применением технологии разреженных матриц. Переход от C2050 (Fermi) к K20c (Kepler) дал прирост производительности на 20%, однако повышение могло быть существенно больше. Но, во-первых, применяется блочное представление разреженных матриц с блоками размером, далеким от оптимума, а во-вторых, уже на C2050 вклад

времени обменов данными хост-GPU был весьма весом. Переход к Kepler увеличивает скорость умножения матриц и уменьшает долю этой операции в общем времени расчета. Доля времени обменов данными хост-GPU превысила 50%. В K20 не реализована анонсированная в прошлом году поддержка PCI-E 3.0, что позволило бы вдвое поднять скорость обмена с GPU.

Прикладными областями, в которых целесообразно применение суперкомпьютеров, являются квантовая химия, нанотехнологии и биотехнологии. Для этих областей актуальны проблемы сокращения времени разработок и получения данных об электронном строении объектов — натурные эксперименты становятся все более дорогими и не дают полной информации об электронных свойствах.

Био- и нанообъекты — это часто сверхбольшие молекулы, требующие для своего моделирования огромных вычислительных затрат. Большие и сверхбольшие молекулы, актуальные для расчетов молекулярной динамикой, чаще всего содержат от тысяч до миллионов атомов. Они имеют гибкую подвижную структуру, легко изменяемую уже при комнатной температуре. Учетом теплового движения и занимается молекулярная динамика, решающая классические уравнения движения Ньютона, например, методом Монте-Карло. При этом возникает необходимость рассчитать энергетический потенциал во многих точках пространства. Когда применяется неэмпирическая молекулярная динамика (потенциал рассчитывается сложными квантово-химическими методами, а не по относительно простым алгебраическим формулам), расчет лимитируется временем вычисления. Такие расчеты являются массовыми, автономными и хорошо распараллеливаются.

При использовании простых потенциалов и применении списков соседей расчет последних часто становится лимитирующим фактором. Молекулярная динамика является приложением, где распараллеливание сдерживается задержками межсоединения, поэтому в кластерах целесообразно применять технологии Infiniband.

Квантовая химия отличается огромным разнообразием методов и алгоритмов расчета молекул, а типичные размеры исследуемых молекул могут отличаться на три порядка (до миллиона атомов и больше). При этом могут кардинально отличаться точность и время расчета, а также возможности распараллеливания. Линейное масштабирование с ростом размера молекулы достигается для сверхбольших молекул на специализированных методах, в частности для DFT и MP2. Подобный быстродействующий приближенный метод DFT разрабатывается и авторами доклада, а соответствующая программа использует для расчетов Nvidia C2050 и K20c.

Программы могут быстро терять эффективность при распараллеливании с ростом числа процессоров (закон Амдала) и накладных расходов на обмены данными между параллельными процессами, следовательно, важны пропускная способность и минимизация задержек межсоединения. Эффективность распараллеливания на большом числе ядер достигается и для большой молекулы, и для вычислительно сложного метода. Высокой эффективности распараллеливания можно достигнуть при применении специализированных методов и программ для расчетов сверхбольших молекул. Метод RI-MP2 разработан для расчетов больших молекулярных систем методом MP2 — в нем используется аппроксимация дифференциальных перекрываний атомных орбиталей. Метод позволяет ускорить расчеты в 40 и более раз по сравнению с MP2, что видно, например, на 240 ядрах.

Метод фрагментных молекулярных орбиталей (FMO) предназначен для гигантских молекул и является надстройкой для других квантово-химических методов. Он основан на разбиении молекулы на отдельные фрагменты — рассчитываются n фрагментов, затем

n(n+1)/2 пар фрагментов при их взаимной ориентации из исходной молекулы. Полная энергия считается аддитивно. Расчеты парных систем и фрагментов автономны и прекрасно распараллеливаются. Однако метод имеет ограничения — во многих сложных молекулярных системах взаимодействие фрагментов неаддитивно (например, в нанотрубках).

Работа поддержана РФФИ (11-07-00470).

http://www.ospcon.ru/files/media/Kuzminsky.pdf

Встроенные многопотоковые приложения для защиты информации

Заборовский В. С., Купреенко С. В. — СПбГТУ

Задачи защиты требуют высокопроизводительных вычислений. В докладе рассматриваются особенности организации параллельных вычислительных процессов, использующих механизмы многопотоковой обработки данных. Для формализации модели вычислений предлагается использовать абстракцию «виртуальное соединение» для формирования характеристической функции доступа к информационным ресурсам. Для вычисления этой функции предложен алгоритм, включающий в себя стадии идентификации, буферизации и параллельной обработки данных с использованием высокопроизводительных многоядерных вычислительных систем.

В рамках предложенного подхода появляется возможность локализации данных и контекста обрабатываемых виртуальных соединений. На примере межсетевого экрана, работающего в режиме скрытной фильтрации, показаны эффективность предложенных решений и возможность повышения производительности и снижения латентности обработки пакетного трафика в нагруженных высокоскоростных компьютерных сетях.

Подготовка специалистов для работы в параллельном мире

Воеводин В. В., Попова Н. Н. — МГУ им. М. В. Ломоносова, Гергель В. П. — ННГУ им. Н. И. Лобачевского

В докладе обсуждаются состояние и перспективы дальнейшего развития системы подготовки кадров, способных эффективно использовать современные высокопроизводительные системы. Основное внимание уделяется итогам выполнения проекта Комиссии при Президенте РФ по модернизации и технологическому развитию экономики России «Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения» за 2010–2012 годы. Представляются основные задачи проекта и приводятся результаты их выполнения. Рассматриваются возможности расширения образовательных инициатив по внедрению параллельных технологий в школьное образование, систему переподготовки профессиональных кадров.

http://www.ospcon.ru/files/media/Popova.pdf

Гетерогенные архитектуры: экосистема для CPU/GPU/DSP

Палташев Т. Т., Перминов И. В. — НИУ ИТМО

Использование графических акселераторов наряду с классическими процессорами является сегодня распространенной практикой в области параллельных вычислений. Подобные вычислительные системы являются гетерогенными — сочетают в себе вычислительные элементы различного типа и архитектуры. Концепция Heterogeneous System Architecture (HSA) представляет архитектуру гетерогенных систем, обеспечивающую общую когерентную память для разнородных вычислительных элементов [1]. Свойство когерентности памяти означает, что различные процессоры видят консистентное состояние общей памяти, даже когда память может независимо обновляться любым из процессоров. Когерентность памяти давно воспринимается как нечто само собой разумеющееся для гомогенных многопроцессорных и многоядерных систем, однако это новая концепция для гетерогенных систем. Поддержание когерентности представляет собой сложную задачу. Но способность процессоров различных типов (CPU, GPU, DSP) работать с одними и теми же данными в общей памяти позволяет избавиться от лишних операций копирования и повысить производительность и энергоэффективность. Приложения в среде HSA, работающие на CPU, могут выполнять отдельные задания на GPU/DSP так же легко, как и на CPU. Для этого приложению достаточно предоставить указатели на данные в общей памяти и обновить соответствующие очереди задач. При традиционном подходе приложение должно собрать все необходимые данные и произвести операции ввода-вывода, задействуя драйверы, чтобы переместить эти данные на вычислительное устройство и запустить вычислительный процесс. Таким образом, HSA позволяет разработчикам ПО писать приложения без необходимости глубоко вникать в специфику работы различных ускорителей, имеющихся на целевой системе, таких как GPU, DSP, видеокодеры/декодеры и прочие акселераторы.

Базовые возможности платформы HSA:

Виртуальная страничная память. Для упрощения работы ОС и приложений на платформе HSA используется единый набор таблиц виртуальных страниц для всех типов процессоров, что позволяет всем вычислителям обращаться в память по одинаковым виртуальным адресам.

Страничные прерывания. ОС позволяет пользовательским процессам работать с объемами памяти, превышающими размер физической памяти, за счет механизма подкачки страниц. Прошлые поколения ускорителей могли работать только с невыгружаемой памятью. Для этого драйвер и ОС должны были выделить участок в физической памяти и создать отдельное виртуальное адресное пространство для акселератора. HSA позволяет избавиться от необходимости использования невыгружаемой памяти и отдельных адресных пространств, позволяя всем акселераторам использовать такое же большое адресное пространство, как и у CPU, с возможностью подкачки страниц.

Очередь задач в пользовательском режиме. HSA позволяет радикально снизить время, затрачиваемое на диспетчеризацию вычислительных задач, позволяя процессам в пользовательском режиме напрямую работать с очередями задач без необходимости вызова системных функций и переключения в режим ядра.

Аппаратный планировщик. Платформа HSA предоставляет акселераторам возможность аппаратного переключения контекста задач, имеющихся в очереди задач, без необходимости вызова функций ядра ОС. Аппаратный планировщик позволяет существенно

ускорить переключение контекста и сократить энергозатраты. Тем не менее ОС сохраняет контроль над работой аппаратного планировщика.

Когерентные регионы памяти. Даже когда CPU и GPU используют одинаковые участки памяти, для традиционных GPU используются отдельные адресные пространства и драйвер графического адаптера должен сбрасывать кэши при совместном использовании памяти с CPU. Платформа HSA предоставляет полностью когерентную память, существенно упрощая написание приложений, использующих такие шаблоны программирования, как производитель-потребитель. HSA также поддерживает реализацию некогерентной памяти, что позволяет получить максимальную производительность в случаях, когда нет необходимости совместного использования данных процессорами разного типа.

Поддержка высокоуровневых языков и технологий параллельного программирования, включая C++ AMP, C++, C#, OpenCL, OpenMP, Java, Python. Бинарный программный код, который может выполняться на CPU и на акселераторах, содержит бинарный код системы команд процессора (CPU ISA) и единый для всех акселераторов HSAIL-код (HSA Intermediate Language). Перед исполнением HSAIL-код преобразуется в код системы команд целевого акселератора.

На текущем уровне развития GPU/DSP акселераторы порой не обладают достаточной гибкостью для решения многих современных вычислительных задач. Архитектура HSA предоставляет доступ к единому адресному пространству со стороны всех вычислительных устройств (для ликвидации лишних операций копирования), механизм управления очередями задач на пользовательском уровне (для минимизации накладных расходов на диспетчеризацию) и переключение контекстов с возможностью вытеснения задач (для улучшения QoS) для всех типов вычислительных элементов. HSA объединяет процессоры CPU, GPU, DSP и другие ускорители в единую систему с общим принципом организации вычислений, что позволяет разработчикам ПО проще решать гораздо большее количество разнообразных задач.

http://www.ospcon.ru/files/media/Perminov.pdf

Литература

1. Палташев Т., Перминов И. Гетерогенная архитектура для CPU, GPU и DSP // Открытые системы. 2013. № 8. С. 12-15.

Облачная образовательная платформа «Персональный виртуальный компьютер»

Соколинский Л. Б., Шестаков А. Л. — ЮУрГУ

Стратегией государственной политики в сфере образования до 2020 года определено, что необходимым условием формирования инновационной экономики является модернизация системы образования. Облачные хранилища данных и облачные вычисления представляют важное направление развития и модернизации современных методик обучения в учреждениях высшего профессионального образования. В докладе рассматривается новый методологический подход к организации обучения студентов с помощью виртуальных рабочих столов, работающих на базе высокопроизводительных вычислительных кластеров.

http://www.ospcon.ru/files/media/Sokolinskiy.pdf

Высокопроизводительная защищенная среда облачных вычислений инженерных и научных расчетов

Заборовский В. С., Лукашин А. А. — СПбГТУ

Компьютерные технологии становятся важнейшим инструментом развития науки, промышленности, транспорта и средств массовых коммуникаций. Новые приложения, востребованные учеными и инженерами, создаются с использованием все более сложных компьютерных моделей, что приводит к возрастанию вычислительной нагрузки, увеличению объема и сложности обрабатываемых данных. В этих условиях важными показателями эффективности становятся масштабируемость сервисов, надежность защиты информации и кластеризация ресурсов для организации высокопроизводительных вычислений. Адекватным ответом на новые технологические вызовы стала концепция высокопроизводительных облачных вычислений, в рамках которой высокая надежность и производительность облачных сервисов достигаются за счет гетерогенности вычислительной инфраструктуры, построенной с использованием современных технологий виртуализации.

Рассматриваемая в докладе платформа ориентирована на широкий класс задач компьютерного инжиниринга — ее архитектура, методы предоставления сервисов и контроля доступа имеют ряд принципиальных отличий как от известных решений для частных облачных систем, так и от публичных облачных сервисов. Разрабатываемая платформа для задач ин-

жиниринга не обладает четко заданным набором сервисов. Для разработчиков масштабируемых вычислительных систем облачная платформа предоставляет сервис по модели IaaS, для инженеров и специалистов в проблемной области (например, прочностные расчеты или газодинамика) — по модели PaaS, а для заказчиков-потребителей вычислительной услуги — по SaaS [1].

Принципиальным отличием инжинирингового центра от классических центров обработки данных является предоставление не только вычислительных ресурсов, но и интеллектуального сервиса в виде специалистов разных проблемных областей. Вокруг вычислительных ресурсов необходимо сконцентрировать группу специалистов: программистов, которые разрабатывают информационные сервисы, инженеров, создающих модели и расчетные алгоритмы, ученых, которые верифицируют и проверяют разработанные вычислительные сервисы. Наличие в инжиниринговом центре экспертов в разных проблемных областях позволит решить комплексную мультидисциплинарную задачу, состоящую из подзадач. Подзадачи организуются в виде «технологической цепочки», компоненты которой разрабатываются специалистами разного профиля: расчетчиками, физиками и математиками. Задачами инженеров инжинирингового центра являются создание средств интеграции разработанных компонентов в единую цепочку, организация решения задачи в рассматриваемой облачной платформе и создание программных интерфейсов сервисов технологической цепочки. Решение задач технологической цепочки происходит в разных сегментах вычислительного комплекса, который объединен в единую вычислительную среду. Компонентами комплекса являются облачный сегмент, высокопроизводительный кластер и специализированные вычислительные системы.

Виртуализация радикально изменила подходы к развертыванию, управлению и использованию корпоративных информационных ресурсов и данных, предоставив новые возможности для повышения уровня масштабируемости числа приложений и консолидации вычислительных ресурсов, выделяемых приложениям. Однако виртуализация привела к тому, что компании столкнулись с новыми угрозами, отражающими возрастающую сложность и динамический характер предоставления ресурсов из облаков. Эти угрозы потенциально могут привести к лавинообразному нарушению безопасности, существенно снижая эффективность традиционных систем защиты информационных ресурсов. Поэтому существующие так называемые подходы Scan-и-Patch, применяемые для контроля уязвимостей, в среде облачных вычислений недостаточны. Традиционные сканеры не могут отслеживать изменения в конфигурации ресурсов в режиме реального времени, что не позволяет точно идентифицировать изменение уровня рисков и принимать меры для блокирования динамически возникающих угроз.

Динамика облачных систем заключается в изменении состава и характеристик виртуальных машин и других вычислительных ресурсов, в миграциях виртуальных машин между гипервизорами и в многопользовательском мультизадачном режиме функционирования облака — система средств защиты должна реконфигурироваться в реальном времени согласно текущему состоянию облачной вычислительной среды.

Отдельно стоит решение по защите облачной среды — это сервис разграничения доступа к вычислительным ресурсам. Главной особенностью облачных систем является динамическая реконфигурация функционирующих в облаке вычислительных ресурсов, поэтому формулировка политики разграничения доступа не должна быть привязана к составу функционирующих вычислительных ресурсов [2]. При этом необходимо обеспечить реконфигурацию системы разграничения доступа в соответствии с текущим состоянием среды облачных

вычислений. Подробно вопрос разграничения доступа в облачных средах рассмотрен в работе [3]. К статическим сегментам применен принцип «сдачи в аренду», который заключается в формировании правил фильтрации для доступа к сегментам только тем пользователям и программным сервисам вычислительной платформы, которые осуществляют решение задачи в данный момент времени. Кроме того, так как в платформах такого типа редки ситуации, когда требуется отдельная виртуальная машина, то было решено осуществлять динамическое создание защищенных сетей, к которым подключаются виртуальные машины, решающие задачи одной вычислительной цепочки. Защищенные сети подключены к межсетевым экранам, интегрированным с программным коммутатором Open vSwitch.

Межсетевые экраны, осуществляющие защиту динамически создаваемых облачных сетей, также создаются в момент инициализации. Важной особенностью применяемого межсетевого экрана является функционирование в безадресном режиме: таким образом, устройство защиты остается невидимым при сетевом взаимодействии и его интеграция не требует реконфигурации сетевой подсистемы облачной платформы. При этом межсетевой экран функционирует

в виде виртуальной машины, что позволяет выделить подсистеме требуемое количество вычислительных ресурсов и при необходимости изменить их конфигурацию. Межсетевой экран осуществляет фильтрацию сетевого трафика на основе правил фильтрации сетевого, транспортного и прикладного уровней, которые сформированы сервисом политики доступа.

Для решения вычислительной задачи, требующей гетерогенных вычислительных ресурсов, необходимо автоматизированное создание защищенного сегмента, состоящего из набора вычислительных ресурсов, объединенного логически. К защищенному сегменту должна быть применена заданная политика безопасности, в которой определена возможность доступа к вычислительным ресурсам владельца решаемой задачи, при этом задействованные вычислительные ресурсы должны быть недоступны другим пользователям инжиниринговой платформы. При завершении задачи результаты должны быть загружены в хранилище данных, а вычислительные ресурсы — освобождены для следующих задач. При этом крайне важно обеспечить разграничение доступа к вычислительным ресурсам, в вычислительной среде может выполняться множество задач одновременно. В докладе рассмотрены способы решения представленных задач. Во-первых, необходимо обеспечить создание группировки виртуальных машин в облачной среде. Для этого использован сервис OpenStack Heat, который поддерживает описание конфигураций в формате Amazon CloudFormation, что также обеспечивает совместимость с публичными сервисами Amazon AWS. Данный сервис позволяет создавать группировки виртуальных машин их заданных образов, виртуальные сети, облачные маршрутизаторы и другие компоненты. Образы виртуальных машин имеют базовый набор сервисов, а расчетные пакеты, прикладное программное обеспечение устанавливаются с помощью инструментария, предоставляемого программным средством Opscode Chef.

Задачи, которые не могут быть решены в виртуальных машинах вычислительного облака, должны быть переданы для решения в специализированных кластерах с использованием высокопроизводительных машин с высокоскоростной коммуникационной шиной, ПЛИС или GPU устройств. Для этого сервис планирования и запуска задач осуществляет резервирование заранее сконфигурированных вычислительных ресурсов, а сервис политики доступа обеспечивает загрузку правил фильтрации для межсетевых экранов, осуществляющих защиту вычислительных сегментов.

В качестве ключевой особенности разрабатываемой платформы необходимо также отметить возможность предоставления суперкомпьютерных вычислительных услуг в виде облачного сервиса. Кроме того, вычислительные услуги, предоставляемые высокопроизводительным кластером, могут быть интегрированы с другими информационными и вычислительными сервисами платформы.

http://www.ospcon.ru/files/media/Lukashin.pdf

Литература

1. Заборовский В., Лукашин А. Высокопроизводительная защищенная облачная среда // Открытые системы. 2013. № 06. С. 10-13

2. Dynamic access control in cloud services. Zaborovsky V., Lukashin A., Kupreenko S., Mulukha V. В сборнике: Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics Сер. «2011 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2011 - Conference Digest» 2011. — С. 1400-1404.

3. Лукашин А.А., Заборовский В.С. Система защиты информационных сервисов в среде облачных вычислений // Информатизация образования и науки. 2013. № 2(18). С. 39-53.

Секция. Стендовые доклады

Экзафлопсы против математического моделирования

Ильин В. П. — Институт вычислительной математики и математической геофизики СО РАН

Переход от петафлопсной к экзафлопсной эре идет поражающими темпами, следуя пока экспоненциальному росту, однако вызывают удивление, по крайней мере, два факта. Первый — никто не может предсказать, к каким качественным новациям приведет эта динамика через 10–15 лет. Второй — наращивание мощностей многопроцессорных вычислительных систем (МВС) идет в основном экстенсивным образом, за счет увеличения количества кластерных узлов с общей иерархической памятью и наращивания их модернизированными графическими ускорителями, огромное количество ядер которых дает высокие отчетные показатели, но архитектурно очень слабо коррелирует со структурой подавляющего числа актуальных сегодня алгоритмов и задач. На удивление мало прорывных компьютерных идей, и, в частности, относительно медленно идет развитие реконфигурируемых методо-ориентированных устройств на базе программируемых логических интегральных схем (ПЛИС), которым, казалось бы, уготованы радужные перспективы.

Мировое вычислительное сообщество идет к осознанию неисчерпаемости потенциала математического моделирования реальных процессов и явлений в производственных и социальных сферах, а также получения новых фундаментальных знаний путем реализации прямых и обратных междисциплинарных задач с обеспечением гарантированной разрешающей способности моделей и алгоритмов за реальное время. Математизация, компьютеризация и интеллектуализация всех отраслей становятся реальностью и в значительной степени определяют научно-технический прогресс, влияя, в частности, на национальную безопасность.

Насущные многочисленные вопросы организации крупномасштабных машинных экспериментов требуют кардинальной смены парадигмы компьютерного моделирования, внедрения последних достижений теоретической математики в вычислительную практику, перехода от уровня палеоинформатики к неоинформатике [1], массового поднятия уровня суперкомпьютерного образования как для роста профессионального контингента специалистов — алгоритмистов и программистов, так и для формирования пользователей новых технологий.

Близок к истине выдвинутый в «дорожной карте» [2] экзафлопсного комитета Дж. Донгарры лозунг о всеобщей смене программного обеспечения и переходе на открытые системы в преддверии прихода компьютеров с миллиардами вычислительных ядер. Существующий в прикладном ПО «зоопарк» образован из коммерческих закрытых пакетов прикладных программ типа ANSYS, NASTRAN или их общедоступных аналогов, а также из многочисленных проблемно-ориентированных библиотек и инструментариев (алгебраические решатели, генераторы сеток, графические системы и пр.), уходящих корнями в однопроцессорные вычисления. Косметическим ремонтом старого программного обеспечения невозможно реализовать эффективное отображение алгоритмов на новые архитектуры МВС.

Настоящее состояние математического моделирования можно определить как кризисное [3] — налицо значительное отставание производительности труда разработчиков прикладного ПО от роста вычислительных мощностей, а также отсутствие эффективного высокопроизводительного ПО для массового использования в решении современных задач на суперкомпьютерах. Существующие программные продукты создаются, как правило, на закрытых принципах, что определяет их разношерстность и многократное дублирование совокупного функционального наполнения. Широкой востребованности математического моделирования препятствует фактическое отсутствие инфраструктуры и технологий организации крупных машинных экспериментов.

Кардинальное решение сложившихся проблем состоит в создании открытой вычислительной интегрированной среды для автоматизации разработки пакетов программ и технологий эффективного математического моделирования. Тенденции создания таких программных окружений для определенных типов приложений уже имеются, например, в работах, выполняемых группами из нескольких университетов. Сюда же относится проект NURESIM — европейская платформа для полномасштабного моделирования ядерных реакторов. В работе предложена концепция и архитектура базовой системы моделирования (БСМ), поддерживающей все основные технологические этапы наукоемкого математического моделирования и ориентированной на согласованное взаимодействие различных коллективов разработчиков и пользователей.

Основные части БСМ — инструментальное ядро и совокупность создаваемых с его помощью программных прикладных пакетов. Ядро состоит из автономных компонентов, каждый из которых включает технологии для «своей» вычислительной стадии: геометрического и функционального моделирования, построения адаптивных неструктурированных сеток, аппроксимации исходных задач сеточными уравнениями различных порядков, решения возникающих линейных и нелинейных алгебраических систем, условной минимизации целевых функционалов в оптимизационных алгоритмах для обратных задач, постобработки и визуализации результатов, принятия решений по итогам расчетов. Информационное взаимодействие компонентов ядра осуществляется внутренними структурами данных (геометрическими, функциональными, сеточными, алгебраическими и т. д.), допускающими множественное представление с конвертерами для обеспечения согласования с внешними продуктами (САПР, графические системы, различные библиотеки). Оперативная разработка конкретных пакетов с помощью инструментального окружения осуществляется на принципах модульного программирования, наподобие конструктора LEGO.

С методической точки зрения условиями успешной широкой эксплуатации БСМ являются: многофункциональность состава современных моделей и алгоритмов для основных типов задач электромагнетизма, упругопластичности, гидрогазодинамики и т. п. с возможностями гибкой программной реконфигурируемости для управления вычислительным процессом, во избежание противоречия между универсальностью и эффективностью, а также для обеспечения различных типов пользователей дружественными интерфейсами; адаптируемость к новым компьютерным платформам и развиваемость состава ядра с обеспечением длительного жизненного цикла формируемых пакетов программ; масштабируемый параллелизм математического обеспечения на всех стадиях численного эксперимента с отсутствием программных ограничений на число степеней свободы и количества используемых процессоров и вычислительных ядер; реализация принципа открытых инноваций с возможностями экспорта-импорта программно-алгоритмических компонентов; наличие представительного

набора тестовых задач и примеров программных реализаций для верификации и наглядной демонстрации эксплуатационных характеристик.

Рассматриваемая структура и проблемная направленность БСМ обуславливают многообразие форм ее использования. Во-первых, ориентация на сверхбольшие задачи делает естественным расчет на облака. Во-вторых, предлагаемая система образует новые возможности кооперации сообщества разработчиков ПО и взаимодействия между разработчиками и конечными пользователями. Это могут быть заказные работы по проведению расчетов или консалтингу, по передаче готовых пакетов или по программированию необходимых модификаций.

По-видимому, индустриальное производство пакетов программ будет распределено между относительно небольшим числом игроков, работающих на поле общедоступных разработок, и создателями традиционных коммерческих продуктов, для которых в рамках облаков актуализируется проблема информационной безопасности. Массовое математическое моделирование действительно может стать третьим по социальной значимости фактором после мобильного телефона и Интернета, но такая перспектива требует еще серьезных технологических продвижений и инфраструктурных решений.

Литература

1. Kleppe A. Software Language Engineering: Creating Domain—Specific Language Using Metamodels.— N.Y., Addison — Wesley, 2008.

2. Ильин В.П., Скопин И.Н. Технологии вычислительного программирования // Программирование. 2011. № 4. С. 53-72.

3. Ильин В. Экзафлопсы против математического моделирования // Открытые системы. 2013. № 05. С. 16-19. http://www.osp.ru/os/2013/05/13035990/

Двумерная виртуализация процессора: новый подход к повышению реальной эффективности суперкомпьютеров

Ефимов А. И. — Конструкторское бюро системного программирования (г. Гомель, Беларусь)

Современные микропроцессоры так же плохи для параллелизма общего назначения, как и динозавры, которые на них охотятся, — для параллельного мира нужны экстремально многонитевые архитектуры, такие как, например, виртуалтредингговая, позволяющая строить эффективные масштабируемые суперкомпьютеры общего назначения для широкого спектра применений. Векторные процессоры Cray положили начало суперкомпьютерам, до конца семидесятых обеспечивавшим требуемую скорость научных расчетов, однако к концу восьмидесятых из-за проникновения компьютеров практически во все сферы человеческой деятельности возникла потребность в объединении векторных и обычных процессоров в масштабируемые вычислительные системы с общей памятью — суперкомпьютеры общего назначения (СКОН) [1, 2]. По утверждению Бартона Смита [3], СКОН должны вывести ИТ-индустрию из спирали специализации, при которой «высокопроизводительными считаются вычисления, хорошо выполняемые лишь высокопроизводительными системами». Такие, «переизобретенные» суперкомпьютеры [3] должны обеспечивать высокую скорость параллельных

вычислений общего назначения, принципиальным свойством которых является обработка интенсивного потока транзакций масштабируемым количеством нитей, взаимодействующих через разделяемую память и интенсивно использующих ввод-вывод с внешними устройствами, включая сетевые.

В докладе предлагается концепция развития многонитевой виртуалтредингговой метаархитектуры (ВТМА), позволяющей устранить Силиконовый Занавес аппаратных тредов — концептуальную причину низкой эффективности современных реализаций параллелизма. Новые системы на основе ВТМА, как и их разработчики и пользователи, способны к длительной эволюции в параллельном мире и могут избежать судьбы динозавров. Вместе с тем для работы с такими системами требуются специалисты, способные вести комплексную разработку аппаратуры ВТ СнК и доработку ОС. Предложенные КПД и метрика реальной скорости могут быть положены в основу технической системы оценки суперкомпьютеров. Сегодня ВТМА пока остается на уровне концепции, эффективность которой сейчас проверяется на макете открытой архитектуры СнК Xilinx MicroBlaze. В случае успеха доказательства эффективности и при наличии конвергенции усилий сообщества российских специалистов может быть создана виртуалтредингговая архитектура на кристалле. Большую практическую пользу может принести виртуалтредингговая реинкарнация системы «Эльбрус-2», что позволит сохранить унаследованное системное и прикладное ПО.

http://www.ospcon.ru/files/media/Efimov.pdf

Литература

1. Smith, J.E. et all, Future general purpose supercomputer architectures. Proceedings of the 1990 ACM/IEEE conference on Supercomputing.

2. Burton Smith, The Quest for General Purpose Parallel Computing 1994, ACM New York, NY, 1994.

3. Burton Smith, Reinventing Computing Microsoft Research Faculty Summit 2007. http://www.cct.lsu.edu/~estrabd/LACSI2006/Smith.pdf

Платформа технологичной разработки математических моделей

Скопин И. Н. — Институт вычислительной математики и математической геофизики СО РАН

Рассмотрение математического моделирования физических процессов и явлений как специально организуемой деятельности для решения прикладных или исследовательских задач приводит к выводу, что эта деятельность нуждается в организационно-технологической поддержке. В качестве основы требуемой технологии целесообразно выбрать жизненный цикл разработки модели и проведения расчетов. Необходима методическая и инструментальная поддержка всех этапов жизненного цикла. Ключевым требованием к реализации идеи является согласованность инструментария — обеспечение корректной передачи информации от этапа к этапу. С учетом множества вариантов решений, которые приходится учитывать при выполнении работ этапов, согласование становится весьма трудоемкой задачей.

Цель любой технологии, в частности технологии моделирования, — повышение эффективности автоматизируемой деятельности. Эффективность связывается с предоставлением разработчику визуальных средств отслеживания выполняемых работ, с указанием

вариантов решений с условиями предпочтения вариантов, с показом связей между этапами — их зависимостями по данным и компонентам. Визуализация помогает корректно упорядочивать работы, планировать контрольные точки и определять готовность результатов. Визуализированные связи регламентируют процесс разработки, определяют допустимые для проекта операционные маршруты деятельности разработчиков. Однако сама по себе визуализация этапов, работ, связей и других элементов проектной деятельности не обеспечивает технологичность. Все компоненты визуального представления проекта (блоки, изображающие этапы, работы, связи передач результатов и данных между работами) должны иметь реальные образы в автоматизируемой деятельности и ее атрибутах: статус, требования к исполнителям, ссылки на документы и пр. Для моделирования эти сущности определяются для каждого этапа деятельности разработчика. Сначала это задание геометрии, свойств объекта моделирования и функциональных зависимостей; затем — дискретизация непрерывной модели, сеточная декомпозиция и аппроксимация функций системой уравнений; далее — решение системы, постобработка и вывод результатов для анализа, выясняющего, соответствуют ли результаты целям моделирования, и определяющего требуемые корректировки пройденных этапов для повторения моделирования, начиная с дефектного этапа.

Универсальные этапы моделирования — это лишь общая структура работ, выполняемых при построении модели и произведении расчетов. Кроме этого, требуется определение работ этапов и их альтернатив, зависящих от текущих результатов. Технологическая поддержка подобных уточнений жизненного цикла заключается в обеспечении накопления решений в виде схем реализации типовых задач.

Повторяемость процесса требует поддержки хранения результатов и их сопоставления при разных итерациях. Она используется как для устранения дефектов, так и при организации серий расчетов, отражающих варианты условий и факторов, влияющих на результаты, в частности, для обратных задач, постановка которых предполагает поиск оптимальных решений. Таким образом, сохранение результатов и истории их получения — существенный элемент технологии моделирования.

Каждый из шагов моделирования предполагает использование методов выполнения работ с подходящими инструментами, выбираемыми исходя из особенностей задачи. Этот выбор должен обеспечить согласованность инструментов с данными, но он далеко не всегда однозначен. Неоднозначность преодолевается специальными проверками критериев допустимости и предпочтительности в зависимости от параметров: специфики данных, истории предыдущих действий и др. Согласование по данным достигается за счет спусковых предикатов, определяющих корректность использования средства для обработки конкретных данных. Если условие не выполнено, проверяется возможность приведения данных, передаваемых на очередной этап, к требуемому формату. Система поддержки согласования должна предоставлять подходящие конвертеры данных.

Обсуждаемая в работе поддержка может повысить эффективность деятельности разработчика только тогда, когда она представлена вместе с предписаниями и регламентами, организующими действия деятельности. Технологичная поддержка не сводится к комплекту инструментов, полезных при выполнении этапов, — она должна содержать фреймворки, которые явно организуют процессы решения типовых задач. Фреймворки накапливаются по мере развития системы, тем самым расширяя сферу адекватного применения технологии [1, 2]. Набор фреймворков, включающих как необходимые для моделирования инструменты, так и средства проверки их применимости, а также средства

ввода, хранения, визуализации, сравнения, конвертирования данных и вывода результатов, образуют платформу, на базе которой организуется технологичная разработка моделей и проведение расчетов.

Платформа предназначена для организационной поддержки деятельности, связанной с моделированием. Понятно, что любая такая поддержка не будет востребованной, если платформа не обеспечит вычислительную эффективность. В первую очередь это касается модельных расчетов. Необходимо, чтобы при заданной точности они занимали приемлемое, преимущественно минимально возможное время и продуктивно использовали процессорные ресурсы. Повышенная эффективность важна и при построении моделей. Обычно эффективность алгоритмов и программ обеспечивается оптимизирующими компиляторами и пользовательским распараллеливанием вычислений. Это приносит определенные плоды, но если для построения моделей такие методы можно считать приемлемыми, то для проведения расчетов их возможностей явно не хватает. Дело в том, что большинство применяемых сегодня алгоритмов разработано в рамках последовательных императивных вычислений, которые с трудом перестраиваются для максимально полного использования всех доступных вычислительных ресурсов. Часто это просто невозможно. Недостаточно развиты и методы оптимизации, использующие информацию о перерабатываемых данных. Эта проблема будет усугубляться с появлением экзафлопсных вычислителей.

Выход из положения в расширяемости платформы. Пополнение ее алгоритмами, программы которых обеспечивают использование процессорных ресурсов по максимуму, в состоянии постепенно адаптировать платформы к развивающимся вычислительным средам. Для этого необходимы стандарты присоединения компонентов, цель которых обеспечить согласованность расширения с уже представленными в платформе средствами. Стандартизация позволит сделать платформу открытой для развития сторонними разработчиками методов, алгоритмов и программ.

Платформа технологичного моделирования предусматривается проектом Базовой Среды Моделирования (БСМ), развиваемым в Институте вычислительной математики и математической геофизики СО РАН, в качестве интегрирующей инструментальной системы для разработки моделей. Концептуальным прототипом ее является Workflow Designer — часть платформы UGENE, предлагаемой для исследований в области молекулярной биологии [3]. Хотя операционные маршруты пользователей-биоинформатиков и разработчиков моделей принципиально различны и прямой перенос прототипа на платформу моделирования невозможен, требования к организационно-технологической поддержке вычислений в этой области аналогичны тем, которые предъявляются к математическому моделированию. Наиболее близкой к БСМ по области применения, предлагаемому инструментарию и методике расширения системой является интеграционная платформа Salome [4]. Принципиальное отличие в том, что инструменты Salome даются без условий их адекватного применения. Отсутствие связи инструментов с особенностями решаемой задачи и с работами, предполагаемыми жизненным циклом моделирования, приводит к тому, что для Salome пришлось разрабатывать обширную документацию, которая недостаточна для самостоятельной работы пользователя, когда он сталкивается с нестандартными случаями.

Представленная в работе платформа развивается эволюционно. Для отработки концепций строится прототип БСМ, в котором поддержка жизненного цикла реализуется фрагментарно. В первой очереди системы представлена поддержка наиболее ресурсоемких компонентов среды — в частности, алгоритмы решения сверхбольших систем линейных уравнений специального вида. Предлагаются максимально масштабируемые

программы для этих алгоритмов, отрабатываются стандарты оформления проверок данных и спусковых предикатов, добавления новых компонентов и другие общие для системы средства.

Литература

1. *Ильин В.П., Скопин И.Н. Жизненный цикл математического моделирования и технология вычислительного программирования // Труды НПО 2011. Рабочий семинар «Наукоемкое программное обеспечение». Новосибирск, 2011. С. 110-116.*

2. *Ильин В.П. Экзапроблемы математического моделирования // Вестник ЮУрГУ. Сер. Математическое моделирование и программирование. 2010. Вып. 6. 35 (211). С. 29-40.*

3. *Грехов Г.А., Скопин И.Н. Языковые средства организации вычислений в области биоинформатики // Системная информатика. 2013. № 1. С. 49-62.*

4. *Salome: the Open Source Integration Platform for Numerical Simulation // Официальный сайт. URL: http://www.salome-platform.org*

Архитектурно-независимая разработка параллельных программ

Легалов А. И. — Сибирский федеральный университет

При разработке параллельных программ, помимо написания логически корректного кода, важную роль играет предотвращение ошибок, обуславливаемых как особенностями решаемой задачи, так и взаимодействием процессов. Это происходит из-за специфики параллельного программирования, ориентированного на создание эффективно работающих программ для конкретных, уникальных параллельных вычислительных систем (ПВС). Несмотря на то что разработчики различных ПВС говорят о легкости создания параллельных программ (например, путем вставки нескольких директив в последовательные программы), реальная ситуация весьма далека от идеала. На различных конференциях, наряду с этими рекламными заявлениями, достаточно часто звучат доклады о трудностях параллельного программирования, при котором зачастую необходимо учитывать тонкую специфику используемых компьютеров, такую как размер оперативной памяти, объем кэш-памяти, особенности системы коммутации и многое другое. Все это превращает применение современных высокоуровневых языков программирования и разработанных с их использованием библиотек в подобие архитектурно-зависимых ассемблеров. Ситуация еще более усугубляется, когда наряду с разработкой параллельных программ встают задачи их тестирования, верификации и отладки. Разнообразные методы приходится комбинировать в различных сочетаниях.

Если сравнить параллельное программирование с последовательным, то можно отметить, что подобное разнообразие архитектурно-зависимых языков существовало десятки лет назад. Широко применялись ассемблеры, а использование языков высокого уровня считалось неэффективным. Сегодня положение в последовательном программировании изменилось коренным образом — ассемблеры используются в достаточно узких аппаратно зависимых областях, а подавляющее число системных и прикладных программ разрабатываются на универсальных языках и с использованием языков сценариев.

Текущее положение дел в области параллельного программирования заставляет заняться поиском подходов, которые бы в идеале обеспечили архитектурно-независимую разработку параллельных программ. Для этого необходимо пересмотреть технологии их создания,

тестирования, отладки и верификации, осуществить ориентацию на модели параллельных вычислений, в минимальной степени зависимые как от ресурсных ограничений ПВС, так и произвольных воздействий со стороны программиста на эти ресурсы и процесс управления параллельными вычислениями.

Произойдет ли переход к подобным универсальным языкам программирования в области параллельных вычислений? Когда и как это может случиться? Окончательного ответа на эти вопросы пока нет, однако работы в направлении создания архитектурно-независимого параллельного программирования имеет смысл вести уже сейчас. Для написания архитектурно-независимых параллельных программ предложен язык функционально-потокового параллельного программирования Пифагор [1], обладающий рядом особенностей: ориентация на бесконечные вычислительные ресурсы; запуск любой операции осуществляется по готовности ее данных и определяется в соответствии с аксиоматикой языка и его алгеброй преобразований; программа не имеет циклов, а значит, и механизмов описания повторного использования ресурсов; для повышения эффективности при описании параллелизма используются специальные программо-формирующие структуры данных, определяемые как списки различного вида.

Функционально-потоковая модель параллельных вычислений имеет ряд особенностей, несвойственных современным системам параллельного программирования: управление по готовности данных с использованием неограниченных ресурсов; распараллеливание программ на уровне элементарных операций; выбор операций и аксиом, определяющих примитивы языка, ориентирован на наглядное текстовое представление информационного графа программы; параллельные вычисления задаются не только копированием потоков данных, но и через разнообразные операторы группировки данных, обеспечивающие динамический параллелизм. Базовые функции модели и языка определены как операторы. Вершины графа, соответствующие операторам, обеспечивают преобразования данных, их структуризацию и размножение. Динамика выполнения операторов задается механизмом продвижения начальной разметки графа по дугам, что соответствует обработке исходных данных, получению результатов и их использованию в дальнейших вычислениях.

Распространение разметки по графу описывается следующими правилами: правилами выполнения переходов между операторами, правилами срабатывания операторов, правилами выполнения оператора интерпретации над предопределенными функциями, правилами эквивалентных преобразований вершин (операторов) и дуг (связей). Правила срабатывания описывают формирование разметки на выходных дугах для каждого из ранее описанных операторов.

Вместо переменных в языке используются обозначения информационных связей по принципу единственного присваивания, которые в дальнейшем используются в различных частях программы для копирования в эти точки связанных с ними значений.

Для проведения экспериментов с языком функционально-потокового параллельного программирования и для развития методов архитектурно-независимого параллельного программирования были разработаны необходимые инструменты: транслятор, простой интерпретатор программ и отладчик. Работы развиваются по следующим направлениям: разработка компилятора с языка функционально-потокового параллельного программирования во внутреннее представление [2]; создание генератора управляющего графа по полученному в результате компиляции реверсивному информационному графу (УГ) [2]; разработка событийного компьютера, используемого для выполнения функционально-потоковых параллельных программ [3]; разработка методов и средств оптимизации РИГ и УГ, аналогичных

методам оптимизации, используемым в компиляторах современных языков программирования [4]; разработка методов верификации функционально-потоковых параллельных программ. После решения всех этих задач планируется приступить к разработке генераторов кода для современных архитектур ПВС, что логически вытекает из целей работы — создать средства, обеспечивающие разработку архитектурно-независимых параллельных программ, а затем использовать их для сжатия параллелизма и наложения ресурсных ограничений.

Литература

1. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. № 1 (10). С. 71-89.

2. Легалов А.И., Савченко Г.В., Васильев В.С. Событийная модель вычислений, поддерживающая выполнение функционально-потоковых параллельных программ // Системы. Методы. Технологии. 2012. № 1 (13). С. 113-119.

3. Редькин А.В., Легалов А.И. Событийное управление выполнением функционально-потоковых параллельных программ // Научный вестник НГТУ. 2008. № 3 (32). С. 111-120.

4. Васильев В.С., Легалов А.И., Постников А.И. Особенности преобразования хвостовой рекурсии в функционально-потоковом языке параллельного программирования // Системы. Методы. Технологии. 2013. № 3(17). С. 106-111.

Многоядерный тупик — есть ли решение?

Любченко В. С. — Александровский завод «РАДИОПРИБОР»

В общем случае ни многопоточные вычисления, ни многоядерность не имеют к параллелизму какого-либо отношения — это некие структурные модели паллиативного параллелизма. На текущий момент нет общепризнанной универсальной модели параллельных вычислений, а развитие параллельного программирования фактически зашло в тупик, один из подтверждающих примеров — безуспешное решение проблемы автоматического распараллеливания.

Нужно ли вообще автоматическому распараллеливанию уделять большое внимание? Очевидно, что параллельные алгоритмы, созданные с нуля, должны быть как минимум не менее эффективны их автоматически распараллеленных аналогов. Вера в возможность автоматического распараллеливания порождает неоправданные ожидания того, что программирование может остаться привычным — последовательным. Думать так — удобно, но рассчитывать — опрометчиво. Несмотря на долгую историю многопоточного/многоядерного программирования, мы способны распараллеливать лишь на небольшое число процессоров — видимо, не число ядер определяет способность решать параллельные проблемы. Если программирование не поменяет своей сути, то огромное число задач останется за бортом параллельного программирования.

Математическая модель с определением алгебры процессов — необходимое и обязательное условие превращения параллельного программирования из интуитивного процесса не только в технологию, но и в науку. Параллельная модель на базе модели конечного автомата, как модели отдельного процесса, и сети автоматов, как модели процессов, — основной претендент на базовую модель мира параллельных вычислений.

Обратим внимание на область цифровых схем, где любая схема, состоящая из двух и более элементов, является примером параллельной системы. Их проектирование хорошо

формализовано, а проверить работу параллельной модели можно, собрав схему из реальных элементов. Пример простой параллельной задачи рассмотрен на моделировании RS-триггера [1]. Выделение триггера среди множества таких же простых задач объясняется тем, что он содержит в концентрированной форме обратные связи, на анализе и реализации которых «ломаются» нынешние теория и практика параллельных систем.

Среди перечня первоочередных проблем, которые приходится решать, проектируя ту или иную параллельную систему, можно выделить: выбор алгоритмической модели последовательного алгоритма/процесса; выбор параллельной модели множества взаимосвязанных алгоритмов; формулировка алгебры процессов на множестве алгоритмов; выбор тестовой задачи для проверки модели вычислений; создание параллельной модели задачи; создание эквивалентной модели тестовой задачи; сравнение результатов работы эквивалентной и исходной моделей с учетом фактических знаний о работе тестовой задачи.

Имея перед глазами схему RS-триггера, можно приступить к созданию формальной модели. Параллельная модель RS-триггера — простой тест на правильность реализации параллельности. Там, где есть проблемы, триггер не войдет в режим генерации (а он, как следует из анализа, должен быть). Одновременно время переключения триггера может служить оценкой эффективности механизма реализации параллелизма. От прохождения «теста RS-триггера» будет зависеть правильная работа не только таких моделей, как клеточные автоматы, нейронные сети — модели, связанные напрямую с автоматами, но и многих других параллельных моделей, применяемых при реализации весьма объемных и сложных прикладных задач.

Создание автоматной модели триггера и ее анализ убеждают, что формализация мира параллельных программных вычислений реальна. Безусловно, есть другие параллельные модели, но в них часто нет той математики, которая есть у автоматов. Например, проблема автоматического распараллеливания, которую безуспешно решают уже длительное время программисты, фактически уже решена в рамках канонического синтеза теории цифровых схем.

Число проблем в параллельном программировании будет множиться до тех пор, пока параллельное программирование остается миром без арифметики и уж тем более алгебры. Базирующаяся на строгой теории автоматная модель вычислений является источником более совершенных процессорных архитектур и параллельных языков программирования: автоматная сетевая модель адекватна любым реальным процессам; алгебра структурных автоматов позволяет «вычислять» поведение параллельных систем; автоматная модель — источник новых параллельных языков и аппаратных архитектур.

На текущий момент технология автоматного программирования — это достаточно обсуждаемая тема даже в обычном — последовательном — программировании. И уже при современном уровне развития программирования и архитектур процессоров нет препятствий для ее превращения в практически доступную и эффективную универсальную технологию проектирования параллельных программных систем.

Выход из «многоядерного тупика» есть [2]. Параллельная автоматная модель уже имеется [3, 4]. Она не отрицает, а во многом упрощает и переводит уже нашедшую свое признание современную архитектуру процессоров на качественно новый — теоретически обоснованный — параллельный уровень. Упомянутая задача моделирования RS-триггера должна рассматриваться как одна из краеугольных задач теории и практики параллельных систем. Само

ее включение в состав подобных задач обосновывается тем, что триггер, как «лакмусовая бумажка», позволяет отделить модели и практики параллельного программирования, которые таковыми могут быть, от тех, которые так лишь называются.

Литература:

1. Любченко В.С. Фантазия или программирование? // Мир ПК. 1997. № 10. С.116-119.

2. Любченко В. Многоядерный тупик: выход есть // Открытые системы. 2013. № 08. С. 52-53.

3. Любченко В., Тяжлов Ю. Осторожно: многоядерный процессор // Открытые системы. 2007. № 6. С. 22-24. http://www.osp.ru/os/2007/06/4337893/

4. Любченко В.С., Перфилов С.А., Ломакин Р.Л. Система управления прессом «АСУ ТП - Пресс» на базе автоматной модели параллельных процессов: свидетельство о государственной регистрации программы для ЭВМ № 2012615341. Зарегистрировано 14.06.2012.

Содержание

Секция. Практика и кадры суперкомпьютерной индустрии

Секция. Стендовые доклады

Тезисы докладов
Четвертого Московского суперкомпьютерного форума

Москва, 2013, 48с.,
Формат 60x90 1/16. Печ. л. 3,25
Тираж 450. 26.11.2013

Отпечатано в типографии
ЗАО "Новые печатные технологии"
тел.: + 7 (495) 223-92-00
info@web2book.ru, www.web2book.ru